MICRO™

THE 6502 JOURNAL

No. 29 OCTOBER 1980 $2.00

Announcing "Hellfire Warrior", a fantastic new Dunjonquest™ computer game... that's really not for everybody: Beginners are likely to be gobbled up in the first room...and there are over 200 rooms on four levels

"Hellfire Warrior." Really not for everybody: newcomers to Dunjonquest should begin with something easier. Here the monsters are deadlier, the labyrinths more difficult, the levels far more challenging...

But for the experienced Dunjonquest game player there are more command options, more potions (13!), more magical items (including—at last—magical armor), more special effects, more surprises. And an innkeeper, an armorer, apothecary and magic shops.

In part a sequel to The Temple of Apshai, up until now the greatest of all the Dunjonquest games, Hellfire Warrior can also be played completely on its own.

Now the character you've created, representing the highest level of role-playing to date, can explore the four new lower levels:

Level 5—"The Lower Reaches of Apshai." With the giant insects and other nasties that swarmed through the upper levels of Apshai. With rooms your hero can get into, but not out of.

Level 6—"The Labyrinth." The only exit is hidden within the Labyrinth. And man-eating monsters can thwart your hero.

Level 7—"The Vault of the Dead."...And of the undead—skeletons, ghouls, mummies, specters... invisible ghosts—lurking in the rooms, doors, secret passages, ready to reduce your hero to a pale shadow of himself. Permanently.

Level 8—"The plains of Hell." In an Underworld of lost souls and shades of dead, of dragons and fiery hounds, of bottomless pits and blasts of hellfire, our hero must rescue the beautiful warrior maiden lying in enchanted sleep within a wall of fire. And bring her past unbelievable dangers and monsters... even Death itself...to sun and air and life itself.

Hellfire Warrior. The most exciting game yet from Automated Simulations, the leading producer of computer fantasy games.

Guaranteed: If it's not the most exciting computer game you've played, return it within ten days for a complete refund.

Available on disk for the Apple II and Radio Shack's TRS80, or on cassette for the TRS80 and for the Commodore PET. **The cassette: $24.95; the disk: $29.95.** Complete with a magnificent instruction manual. Some of the drawings in the manual are reproduced here (in greatly reduced scale).

Use the handy coupon or, if you wish to order by MasterCard or VISA, use our toll-free phones: In the United States: operator 861 **(800) 824-7888**; In California: operator 861 **(800) 852-7777**; In Hawaii and Alaska: operator 861 **(800) 824-7919**.

AUTOMATED SIMULATIONS, INC.
P. O. Box 4247, Dept. DH Mountain View, CA 94040

☐ Yes, I want Hellfire Warrior. On a satisfaction-guaranteed/full-money-back policy. Please send me the Hellfire Warrior on:

_____	**Disk** for TRS-80 (32K, TRSDOS) @ $29.95	$ _____
_____	**Disk** for Apple (48K with Applesoft in ROM) @ $29.95	_____
_____	**Cassette** for TRS-80 (16K, Level II) @ $24.95	_____
_____	**Cassette** for Commodore PET (32K, old or new ROMS) @ $24.95	_____

Sub Total $ _____
Plus shipping & handling $1.00 each _____
*Plus sales tax for California residents _____
TOTAL $ _____

☐ I enclose my check in the amount of $ _____
☐ Please charge to my ☐ VISA or ☐ MasterCard account
My card no. _____ Expires _____
Signature _____
Name _____
Address _____
City/State/Zip _____
*California residents: please add 6% or 6½% sales tax as required.

☐ **Please send me your complete catalogue of computer and board fantasy games.**

LOWER CASE +PLUS

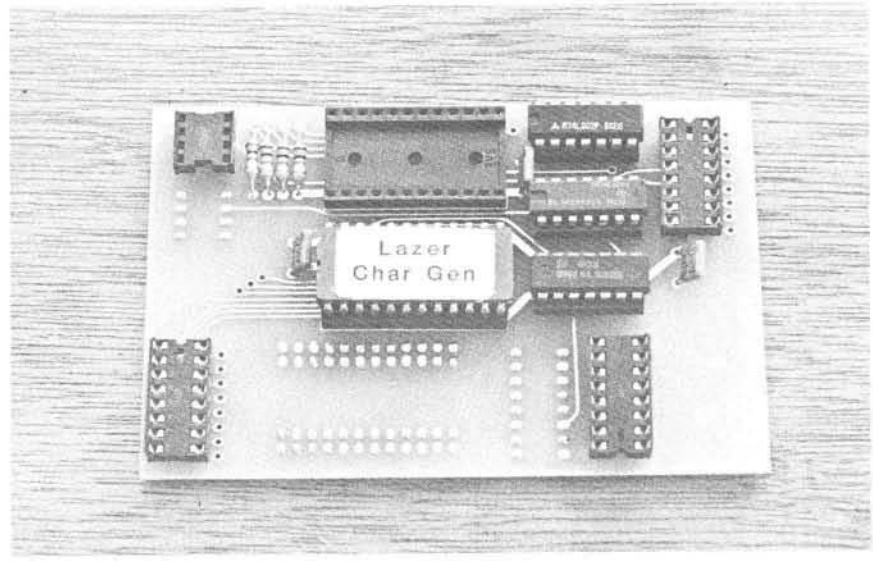
by LAZER SYSTEMS
for the Apple II

$59.95

at better computer stores everywhere. Or, order direct from us.

Calif. residents add 6% sales tax.

Allow 3 weeks for delivery. Outside continental U.S.A. shipping charges will be added.
Dealer inquiries invited.



CHECK THESE FEATURES:

LOWER CASE CHARACTERS:
Put normal and inverse lower case characters on the Apple II text screen.

CAPATIBILITY:
The Lower Case +Plus is compatible with word processors and text editors that utilize a lower case adapter to put lower case on the screen.

RESET DISABLE:
You can disable the reset key or control the reset disable with external switching through the expansion socket.

2 CHARACTER SETS:
There are two complete character sets, on board, in the Lazer Systems exclusive character generator. The first set is a word processing set and the second is a HI-RES graphics set. Character set selection is accomplished through the expansion socket.

SOFTWARE:
Standard BASIC software comes on diskette with each Lower Case +Plus. Pascal software is available for $9.95 extra.

2716 EPROM:
The Lazer Systems character generator is pin for pin compatible with the 2716 EPROM. This will allow you to design you own character set for use with this board. The character font is compatible with Mnt. Hardware's ROM+ and other character generator programs to make it easy for you to build your own character set for the 2716 EPROM.

GRAPHICS:
You can do extremely fast HI-RES graphics on the text page by using graphics character set or by purchasing the GRAPHICS +PLUS board listed below. This does not require the use of the 8K HI-RES page.

EXPANSION:
The on board expansion socket will let you control the reset key disable and character set selection with either a hardware switch or software control. It will also allow you to have an external character set in RAM that you can control & re-define with software.

DOCUMENTATION:
You get a 12 page documentation booklet to show you how to control the fantastic capabilities of this board.

COMING PRODUCTS TO SUPPORT THE LOWER CASE +PLUS:

KEYBOARD +PLUS:
This board will plug directly onto the Lower Case +Plus and will allow you to control the upper-lower case with the shift key as well as giving you a hardware keyboard buffer that will allow you to enter commands and other inputs while the processor is out handling other operations such as disk 1/0. Look for it in December 1980.

GRAPHICS +PLUS:
This graphics board will take full advantage of the Lower Case +Plus board's expansion socket. It will allow software controle of the reset disable, character set selection and control of a RAM based character set. With this board, you will be able to do extremely fast HI-RES graphics on the text page simply by defining and re-defining the character set that is provided in RAM on this board. Look for it in February 1981.

P.O. Box 55518 • Riverside, Ca 92517

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™.

MICRO™

October 1980
Issue Number 29

MICRO™ is published monthly by:
MICRO INK, Inc., Chelmsford, MA 01824
Second Class postage paid at:
Chelmsford, MA 01824
Publication Number: COTR 395770

Subscription rates:U.S.	$15.00 per year
Foreign surface mail	$18.00 per year
Central America air	$27.00 per year
So. Amer/Europe air	$33.00 per year
Other air mail	$39.00 per year

For back issues, subscriptions, change of address or other information, write to:
MICRO
P.O. Box 6502
Chelmsford, MA 01824
or call
617/256-5515

Articles

Departments and Sundry

BPI™

GENERAL ACCOUNTING SYSTEM

The Standard For Small Business Accounting ™

GENERAL LEDGER SYSTEM
BALANCE SHEET
PROFIT & LOSS STATEMENT
CASH DISBURSEMENTS JOURNAL
(Check Register)
SALES JOURNAL
(Invoice Register)
CASH RECEIPTS JOURNAL
MERCHANDISE PURCHASED JOURNAL
CASH SALES JOURNAL
GENERAL JOURNAL
ACCOUNTS RECEIVABLE LEDGER
ACCOUNTS PAYABLE LEDGER
PAYROLL LEDGER

ACCOUNTS RECEIVABLE SYSTEM
OPEN ITEM OR BALANCE FORWARD
COMPLETE ACCOUNT AGING
LIFE-TO-DATE CUSTOMER HISTORY
COMPLETE ACCOUNT ANALYSIS
(Up to 16 Reports)
ITEMIZED STATEMENT PRINTING
CUSTOMER MAILING LIST GENERATION
500 PROMPTED CUSTOMERS PER DISKETTE
(No Limit to Number of Diskettes)
1000 TRANSACTIONS PER PERIOD
SKELETON GENERAL LEDGER PREPARED
AUTOMATIC INTERFACE WITH BPI INVENTORY SYSTEM
AUTOMATIC POSTING TO BPI GENERAL LEDGER

INVENTORY/ORDER ENTRY
FIFO, LIFO, AVERAGING
900 - 2000 LINE ITEMS
INSTANT INVENTORY QUERY
AUTOMATIC BACK-ORDER HANDLING
AUTOMATIC RE-ORDER LEVELS
INVENTORY TURN ANALYSIS
POINT-OF-SALE OPERATION
PRICE TAG PRINTING
INVENTORY PROFIT & LOSS STATEMENT
PURCHASE ORDER GENERATION
NON-INVENTORY ITEM HANDLING
ALWAYS IN BALANCE WITH GENERAL LEDGER
AUTOMATIC INTERFACE WITH BPI A/R SYSTEM
AUTOMATIC POSTING TO BPI GENERAL LEDGER

Why do we claim the BPI SYSTEM to be the STANDARD for SMALL BUSINESS ACCOUNTING?

Because it was designed BY BUSINESS and ACCOUNTING PROFESSIONALS—not programmers.

It lets a business run the way it DOES—not the way a group of programmers THINKS it should.

Why is it the MOST POPULAR Small Business Accounting System on the market today?

Because it is FAST, EASY TO USE, FULLY INTEGRATED, DESIGNED FOR THE FIRST-TIME USER and SUPPORTED and SUPPORTED.

SUPPORTED by BPI SYSTEMS and SUPPORTED by SOLUTIONS, INC.

SUPPORTED by BPI SYSTEMS to assure you THE SYSTEM will be IMPROVED and EXPANDED to give you the most complete and useable accounting system available at any price.

SUPPORTED by SOLUTIONS, INC. to assure you of fast update materials, the latest information and a PROFESSIONAL approach to APPLICATION Marketing and Distribution.

DESIGNED FOR MICROCOMPUTERS MANUFACTURED BY —

APPLE COMPUTER BELL & HOWELL
COMMODORE

FEATURES

- Chart of Accounts — over 500.
- Up to Ten separate profit centers, each with a complete set of financial statements—PLUS a set of consolidated statements.
- Accounts Payable — over 200 PROMPTED payees.
- Accounts Receivable — over 500 PROMPTED customers per diskette.
  1000 transactions per period.
  Up to 16 REPORTS for ACCOUNT ANALYSIS.
- Inventory — From 900 to 2000 line items.
- All sections may be used independently.
- Designed for APPLE, BELL & HOWELL and COMMODORE microcomputers.
- THOUSANDS IN USE today by BUSINESSES, C.P.A.'s and BOOKKEEPERS.

"...approved by Arthur Anderson, the accounting firm, and has been accepted by Commodore as its factory-approved accounting system."

VENTURE Magazine, September 1980

AVAILABLE AT YOUR DEALER

DESIGNED AND DEVELOPED BY

—BPI SYSTEMS, INC.—

A unique combination of Accounting, Business and Programming expertise dedicated to making the BPI GENERAL ACCOUNTING SYSTEM the finest system in use throughout the world...

THE STANDARD FOR SMALL BUSINESS ACCOUNTING ™

DEALER INQUIRIES INVITED
Distributed By

SOLUTIONS, INC.

3740 Colony Drive
San Antonio, Texas 78230
(512) 690-1017
The Source: TCI 170

Software Distribution

This editorial is in two parts. Part 2 will appear next month.

Part 1

You have written a piece of software you think is pretty good and which you feel would be of interest to other computerists. You decide that you would like to distribute it in some manner for three reasons: 1. you are a "nice guy" and want to help your fellow computerists; 2. you would like to get some credit and recognition for the good programming that you have done; and 3. you would like to make a few dollars. Now the problem arises: how do you distribute this software most effectively to serve the above purposes? There are a number of options, and I will cover them here.

Give it away. Your product may not be marketable for money for a variety of reasons, or you may simply not care about a limited return on a marketing or sales effort. Software which would not be marketable would include that which is too limited in interest and/or application, such as a driver for a very uncommon piece of peripheral equipment; a game or demonstration program which is not significantly different from other products on the market; a utility which is too small to be a saleable item; or a program which would cost too much of your time and effort to document to the point required to make it marketable. There are, I am sure, other reasons for *not* selling your software, but, you get the idea. If you choose not to sell your software, you may want to let others use it by giving it away. This may be done in several ways. Donate it to your local computer club library; donate it to a national user group for your microcomputer; send it to a magazine which does not pay for articles; etc.

Sell it as a software package. Assuming that your product has a more than minimal potential market value, what is the best way to make money on it? There are two major routes to selling a software package. First, and most obvious, is to package and sell it yourself. Any software which you plan to sell should be well documented. This is a major stumbling block for many computerists. Programming is fun; documenting is a drag. If you are going to sell it yourself, then you are solely responsible for writing, editing, and printing the documentation, and you must handle sales by mail order and/or through dealers. While this method may have the greatest reward (my company is founded on the profits from a $10.00 demonstration package for the KIM-1), it entails the most work and greatest risk. MICRO tries to aid this type of effort via its free MICRO Software Catalog, and a number of people have been very successful using this service to get started.

The second common sales method is to work though an existing software distributing company. A number of these are connected with major computer magazines such as *Creative Computing* and *Kilobaud Microcomputing*. Others are independent and may or may not concentrate on a particular market segment such as 6502, TRS-80, APPLE, etc. You will normally be expected to provide good documentation, but you may also get some help from the company. All of the risk and effort of marketing are borne by the company and you receive a royalty, usually a fixed percentage of the selling price. This percentage may be quite low. I know of one case where an individual's software was approximately one-ninth of a total package being sold for $150.00, and his payment per copy was 25 cents, which came to approximately 0.16 percent (0.0016 cents on the dollar)! The normal royalty payment will be higher. Be sure you shop around. Prices can vary considerably. Also consider the total distribution. A large-volume distributor may offer you less per copy than a smaller distributor, but may sell many more copies.

Robert M. Tripp

The Sporting Micro
Cover Artist
Terry Allen

The cover depicts, tongue-in-cheek, a micro being used to score a medieval joust. Actually, a micro is ideally suited for scoring and keeping track of our far more complicated modern sports.

Timing: Numerous sports require accurate timing. A micro makes the ideal controller for a basketball clock, automatic race timer, auto race lap-time/speed calculator, and so on. The input may be manual and not require any additional peripheral equipment, or may be generated by various peripheral devices.

Statistics: Many sports enthusiasts now demand more than a simple won/lost type of scoring. They want to know all types of statistics: batting average for the season, against left-handed pitchers, against a particular team; pass-completion percentages, yards-per-carry, total yardage; etc. TV and radio sportscasters provide some of this data, but a micro could permit a fan to call up the information which he is interested in—when he wants it!

Averages and Handicapping: Any league secretary can tell you that keeping the bowling or golf averages and handicaps for a team or league can be a tedious job. How simple for the micro to keep all of the pertinent information updated and instantaneously determine any required set of averages and handicaps.

Scoring: Some sports have complex scoring requirements and could make good use of a micro. The scoring of Olympic ice skating or gymnastics requires certain scores to be dropped, others to be averaged, and other factors to be weighted. Any event which has a large number of participants—such at the Boston Marathon—could obviously use a number of micros to help keep account of who is and who is not on the track.

Summary: A microcomputer can have a number of roles in the sporting world.

An OSI Cheep Print

Here is the answer to the age-old question, "How do I get data from my computer to a remote printer?" Hardware requirements are discussed and software is provided.

Thomas R.Berger
10670 Hollywood Blvd.
Coon Rapids, MN 55433

When I settled on an OSI computer, I knew that OSI's documentation was not the best. The documentation that I received was far better than I had expected, but there were no disassembled listings of machine software. The dealer from whom I purchased the computer was very helpful, but even that is no substitute for hard copy listings. OSI's OS65D operating system contains a disassembler so that listing of a few program lines is possible. Even so, it is very difficult to understand a large program seeing only 20 lines at a time.

The program offered here solved my problem for almost no cost. It will run on Challenger 2, 4, and 8 computers equipped with a disk and OS65D. (I have OS65D V3.2 NMHZ 10/79 POLLED.) The machine language program will run on any OSI computer with the keyboard tone generator. However, the BASIC program in Listing 2 will only connect the routine to OS65D. Changes are needed if you have a non-disk system. The program has been so valuable that using it I have written a BASIC text processor whose output I run on an IBM selectric for beautiful right and left justified copy.

About the time I received my computer, a friend, C.A. McCarthy, told me of a program he had developed to use on his PET to send tones to a tape recorder. These tones were subsequently played back to a printer with attached modem to produce hard copy. I was new to machine programming, so he kindly sketched the crux of such a program for me. After disassembling the operating system and experimenting with possible ways to use the program, the present version evolved.

Recently, Charlie gave me a copy of his article 'Cheep Print' which appears in *Compute* No. 3. Even though it is written for the PET, rather than a Challenger, this article contains valuable information. I will only repeat the more vital data given in that article.

The Equipment

To use this program you need: (1) a cassette tape recorder; (2) a kludge to decrease the volume from the computer output to the recorder; (3) HIFI phonograph cables to connect things together; (4) a telephone handset; and (5) someone else's printer with a modem attached. (You can live without items (2), (3), and (4) if you are a pauper and warbling tones are music to your ears.) Let's run over these items one at a time.

Almost any cassette recorder will do. I am probably unique among computerists in owning a cassette recorder which is unsuitable for computer use. This machine is an older Panasonic. For reasons known only to themselves the designers geared the capstan drive to the capstan roller. Gear enhanced warble does wonders toward destroying the enjoyablility of music and the intelligibility of computer tones. Panasonic must have fired those engineers since newer models do not have this ingenious innovation. I borrowed Charlie's 'Brand X El Cheapo' cassette machine and have since bought a small GE recorder. The printers can't seem to tell the difference between these two because they both work fine.

The schematic for my kludge is given in Figure 1. The potentiometer is nothing special. Anything from 10K ohms on up will do. I have used both 10K and 1 Megohm pots. The 10K choice is slightly easier to adjust. The resistor can have values from 4.7K to 47K and still work; be creative. My kludge is housed in a $1.5 \times 1.5 \times 2.25$ inch metal box. A plastic box or pill bottle should work fine. This kludge is a real handy dandy. When I travel, I often take a recorder with me. It is easy, using the kludge, to dub tapes from the radio, a HIFI, or another recorder. OSI protects the tone output with blocking capacitors. However, if you record music such protection may not be present. If you find the kludge not working, the reason may be due to voltages appearing at the input of your recorder. Inserting the optional capacitor should cure all problems. I have never used such a capacitor and have had no problems. Adjustment of the kludge is always the same: record music or tones while fiddling the pot on the kludge; set the pot at a level slightly below the point where the recording begins to sound awful. That's all there is to it.

The cables are the ordinary kind available at HIFI, electronic, and radio shops. You need ones to fit the holes on your recorder, your computer, and the kludge.

Letterbox

We often get letters, such as the following, which ask for advice. They are frequently long, and for lack of time, we cannot answer then individually, point by point. Furthermore, we are publishers, not consultants, and dare not recommend specific products, despite our personal experience and preferences.

We know, however, that some of our readers could and might like to answer other readers' queries. Therefore, as space permits, we will print such letters here.

NOTE TO THOSE SEEKING ADVICE: To increase your chances of having your letter published—please be brief! The following letter had to be shortened considerably:

Dear Editor:

I am planning to purchase an Apple II but am running into a problem in planning the system I will get. Most computer stores tend to push the brands of hardware they carry. Thus, I don't think I'm getting objective information. In particular, I am vexed by the choice of which printer to get. Money is a big factor, but also of prime interest to me are two uses to which I plan to put the computer: I want to use it in text editing and also to aid me in musical composition with the appropriate hardware/software.

Should I get one printer that does both letter-quality work and the necessary graphics for musical scores, or would I be better off buying two printers, one letter-quality and one with graphics ability? (one dealer suggested the Centronics 737 as being of letter quality; do you believe this print is acceptable?) Second, in line with your answer to my first question, could you recommend a printer or printers that fit my specifications?

David Ben Leavitt
8044 Germantown Avenue
Philadelphia, PA 19118

Dear Editor:

I enjoyed reading the article on "BCD Input to a 6502 Microprocessor", (August 1980 27:68-70) since I recently went through the same problem with an AIM-65. The subroutine which unpacks a pair of BCD digits (lines 500-570 of the program) is unnecessarily complex. If you think hard about the problem, or do as I did, just enumerate all the cases and look for the pattern, you will find that the conversion basically involves longhand division of the input by 16. If the two digit input is XY the quotient is the left hand digit(X) and the remainder is the right hand digit(Y). The BASIC programs listed below will do the conversion. The last line repacks in base 10.

```
I    10 XY = PEEK (40960)
     20 I = XY/16
     30 X = INT(I)
     40 Y = (I-X)*16
     50 XY = (10*X) + INT(Y)
II   10 XY = PEEK(40960)
     20 X = 0
     30 Y = XY
     40 IF Y < 16 THEN 80
     50 X = X + 1
     60 Y = Y-16
     70 GOTO 40
     80 XY = (10*X) + Y
```

Program I is cuter, but program II is probably faster. The roundoff problems discussed in the article preceding the one being discussed don't apply since 16 represents exactly in binary. (I tried all 100 cases on an AIM-65.) It is not clear to me what the fastest way to do the job is if you work in assembler or machine code. Are there any readers out there who would care to explain the fastest way for us?

Richard Garber
3535 Greenbrier Blvd., Apt. 34A
Ann Arbor, MI 48106

The telephone handset is a convenience: it fits into the modem cradle and should have a jack on the other end to plug into your recorder output. A very small transistor radio speaker could probably substitute for the handset. You can buy an expensive telephone at your local discount store and cut the handset off; you can haunt the radio junk shops; or you can touch-up a local radio amateur club for information. There seem to be telephoneless handsets floating around at reasonable prices.

The kludge, wire, and handset are not essential. I did my first printing over the phone lines by holding the phone up to the speaker on my video TV monitor. Later I held the tape recorder microphone up to the TV to record, and the recorder itself up the the modem to play back.

Finally, you need a printer with a modem attached. I use two at work. Students may be able to use printers at their schools. (Be sure your employer or school approves of this procedure.) A buddy may let you use his. If you are looking for a suitable buddy, join a computer club. After using this program you will covet a printer. If your local computer shop realizes this, they may let you use their printer for a small 'paper and use' fee.

These gadgets are connected together as follows: the tones come out of the tone generator on your computer; this output is on the back panel of the C4P and on the back of the keyboard of the C8P. (See your instruction manual.) Use your wire to connect the tone generator to the kludge and then the kludge to the microphone input on you casette recorder. Now it's ready to go. The poor man's approach is to connect the tone generator to the video TV monitor and play it into the microphone on the cassette recorder.

The Programs

Now that we have the nuts and bolts under control let's move on to the programs. The assembly language program appears in Listing 1. The 0S65D operating system has an Assembler. In fact, this program was prepared on the Assembler. This is probably the quickest way to put CHEEP PRINT into your system. Those following this route probably need no further help.

If you're not a programming ace, Listing 2 gives a BASIC program that will: (1) load CHEEP PRINT; (2) ask you for and set all parameters; and (3) start the program running. The various tone mode data are listed in Table 1. A baud rate of 110 is reliable but slow; 300 is more acceptable, but the printer you use may give you no choice about baud rates. The ANSWER or ORIGINATE modes may also be fixed by the modem. The OSI tone generator is more 'fine-tuned' for low tones, and I have had the best luck with these frequencies, even though my first several hundred pages were successfully run with the high tones. You may also prepare Kansas City Standard tapes with this program. The necessary data are given with the modem data.

Those of you who have ROM BASIC without 0S65D are left pretty much on your own, since the BASIC program given here assumes you have a disk drive and 0S65D. If and when I obtain full information on how to connect the routine on non-disk systems, I will publish it. Someone with such a system could solve the problem by writing the solution down and sending it to his favorite computer magazine. The secret is to change the pointers to the printer subroutine so that they point to CHEEP PRINT instead. With ROM BASIC this may be quite a trick.

Using It

Finally, we need to know how to use the program. Once the program is loaded, the computer thinks CHEEP PRINT is a printer. Thus all commands or statements which would cause a printer to print (if you had one) will cause CHEEP PRINT to chirp instead. In a BASIC program,

```
10 PRINT "GARBAGE"
```

will cause the word "GARBAGE" to appear on the screen. In the same way,

```
10 PRINT #4, "GARBAGE"
```

will send the word "GARBAGE" warbling out your tone generator. Similarly LIST #4 would list a program via tones. When you use the device #4 statements, you will see nothing on the screen; it's all going out the tone generator. Don't lose heart in this case because your computer really is working. You may also reset the input/output flags via the operating system I0 command. The command (from BASIC) DISK"I0, 0A will cause all output to go both to the screen and out the tone generator. DISK!"I0, 02" will

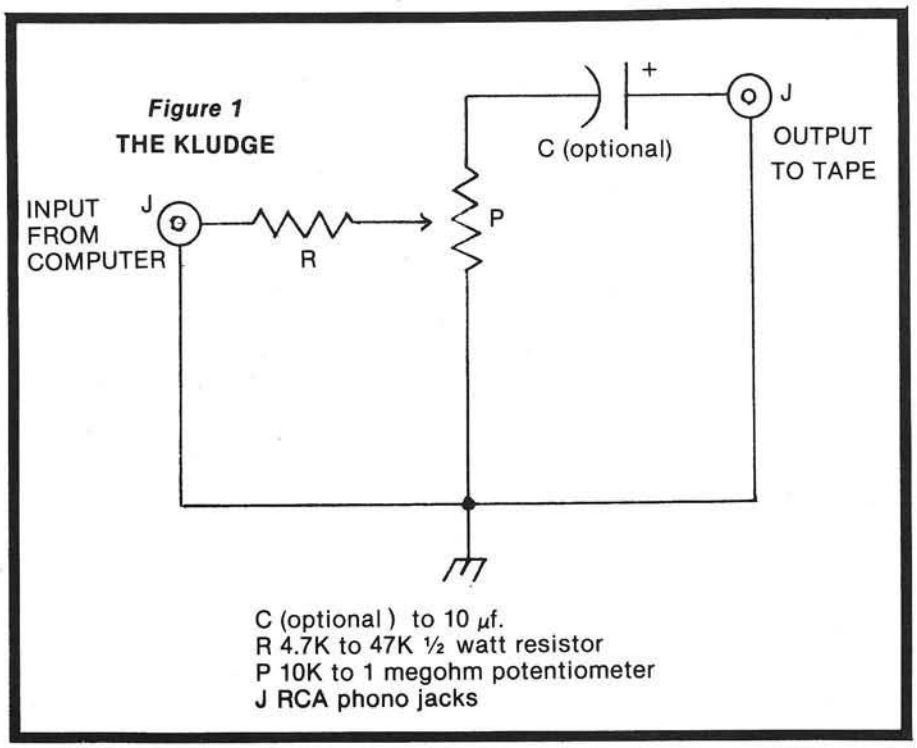


Figure 1
THE KLUDGE

C (optional) to 10 μf.
R 4.7K to 47K ½ watt resistor
P 10K to 1 megohm potentiometer
J RCA phono jacks

reset the I0 to screen only. Beware that the operating system objects to your changing the output flag so that for certain changes in operating mode (such as disk errors or a CONTROL C) it will reset the flag to screen only. If you reset the computer, you will have to reload CHEEP PRINT.

The loading program will ask for a carriage return time. This must be determined experimentally. I use a wait time of 20 for an NCR thermal printer which has a fast carriage return. The IBM selectric is so slow that I not only use full wait time of 255, but also add extra stop bit time equal to 40 after each character. In my word processor program I use a BASIC wait loop for carriage returns. The stop bits in the table are minimum. They provide wait time after each character and may be increased to any number greater than the minimum.

All parameters are set in the beginning of the BASIC loading program in Listing 2. The equations for calculating these parameters are given in the Appendix. There is a great deal of flexibility in the recording formats that you can choose.

The Bottom Line

Now comes the fine print. The OSI tone generator is not perfect and neither is cassette tape. When many appliances are switching off and on, they may upset your tape recorder and cause a slight pop to be recorded on the tape. The keyboard cables which came with my C8PDF were quite short, so I added longer cables. This caused havoc with the tone generator. I still have not fully cured the problem. The computer's pops and clicks are coupled through the long cable to the tone generator causing it to pop and click. Interestingly enough, these pops are most severe in the left margin of CHEEP PRINT output. When I find a sure cure, I will let people know. Modems don't like these pops and say so by causing the printer to botch a letter or two. If you run repeated hard copy and find the same errors in the same places, then the problem is with the cassette recording. Listen carefully to the recording and see if you can hear the errors. One solution is to spend more money for filters. My solution is to record when the appliances are quiet, plug the recorder into a separate outlet, and keep it away from the computer. I also tolerate an occasional error. Before I lengthened the keyboard cables, I obtained almost perfect copy. The few errors I did have could be heard as tape dropout and minor clicks. This article was prepared using the word processor and CHEEP PRINT. I am happy with CHEEP PRINT (even though I wish I had my own printer).

If you find many nonrepeating errors then your tone generator and the modem have a disagreement. You own your tone generator so you change its personality to make it more agreeable. Since not all modems are created equal, you may have different changes for different modems. Listing 3 gives a BASIC program which will allow you to 'fine tune' your tone generator for a particular modem. Run this program and record the output via CHEEP PRINT. Look over the printed output for the 'best values ' of HI and LO. Insert these values in the loading program of Listing 2 for L1, H1, L2, etc. You will probably find that the calculated values are best.

```
                        ;OSI CHEEP PRINT
                        ;T.R. BERGER 3/80
                ;
                ;ASSEMBLY DATA
                ;
                        * = $5000       ;START ADDRESS
0092=           SH = $92                ;DELAY SHORT
0003=           LG = $03                ;DELAY LONG
0004=           SP = $04                ;STOP BIT TIME
0014=           WT = $14                ;CARRIAGE RETURN WAIT
002E=           LO = $2E                ;LOW TONE DIVISOR
0027=           HI = $27                ;HIGH TONE DIVISOR
                ;
                        ;THIS IS THE MAIN PROGRAM
                ;
5000 48         PHA                     ;SAVE CHAR ON STACK
5001 78         SEI                     ;INHIBIT INTERRUPTS
5002 A22E       LDX #LO                 ;SET LO TONE DIVISOR
5004 8E01DF     STX $DF01               ;TURN TONE ON
5007 202E50     JSR DELAY               ;WAIT ONE BIT
500A 38         SEC                     ;SET UP A STOP BIT
500B 6A         ROR A                   ;GET FIRST BIT
500C A22E   LOOP1 LDX #LO               ;LO TONE DIVISOR
500E 9002       BCC BR                  ;GET HIGH TONE IF 1
5010 A227       LDX #HI                 ;HI TONE DIVISOR
5012 8E01DF  BR STX $DF01               ;TURN ON TONE
5015 202E50     JSR DELAY               ;WAIT ONE BIT
5018 4A         LSR A                   ;GET NEXT BIT
5019 D0F1       BNE LOOP1               ;IF MORE BITS THEN
501B B0EF       BCS LOOP1               ;GO BACK & SEND THEM
501D A204       LDX #SP                 ;STOP BIT TIME
501F 203250     JSR LOOP2               ;SEND STOP BITS
5022 68         PLA                     ;GET CHARACTER
5023 C90D       CMP #$0D                ;CARRIAGE RETURN?
5025 D005       BNE DONE                ;NO, THEN DONE
5027 A214   WAIT1 LDX #WT               ;YES, THEN WAIT
5029 203250     JSR LOOP2
502C 58     DONE CLI                    ;INTERRUPTS OK NOW
502D 60         RTS                     ;QUIT
                ;
                        ;THIS IS THE DELAY
                ;
502E A203   DELAY LDX #LG               ;LONG DELAY TERM
5030 A092       LDY #SH                 ;SHORT DELAY TERM
5032 88     LOOP2 DEY                   ;COUNT DOWN Y
5033 D0FD       BNE LOOP2               ;COUNT Y TO 0
5035 CA         DEX                     ;COUNT DOWN X
5036 D0FA       BNE LOOP2               ;COUNT X TO 0
5038 60         RTS                     ;GO BACK
                .END
```

Listing 1

But Mom, It doesn't work

I am annoyed when I enter a program exactly as written and it doesn't work as it's supposed to. I can't promise you this one will work, but if it doesn't, the following hints may help.

There are several levels of 'NOT WORKING.' They are: (1) nothing happens; (2) the system crashes when the program is loaded; (3) the system crashes when you try to output via CHEEP PRINT; (4) the tone comes on but it doesn't warble; and (5) you get nice warbling tones but they look like mish-mash on the printer. The cures are as follows. First, check (character by character) the BASIC loading program in Listing 2. An error in a DATA statement or a POKE will stop the program from running properly. Next get your manual and compare the addresses given there with the ones listed at the end of the APPENDIX. If they don't match, use the ones from your manual. There is one crucial address you will not find: the one for the beginning of the printer subroutine in the 0S65D input/out-

put table. It should be where I say it is, but if it isn't, hop off to your local OSI dealer and ask him, "Where is the OS65D I/O jump table located in my memory map? In this table, where is the printer subroutine address?" You don't want the address of the printer subroutine. You do want the two addresses of the two locations in the jump table which store the address of the printer subroutine. (Isn't that confusing?) These two addresses, converted to decimal, appear as the addresses in the POKEs in line 670 of Listing 2.

If you've managed to wade this far, you should be able to play CHEEP PRINT through your video monitor and hear warbling tones. Now, about that mish-mash: Carefully read your manual on your tone generator. Does everything check with the APPENDIX data? If not, use the values from your manual.

My computer has a 1MHZ clock frequency. All of these programs assume this is the clock speed. If you have a newer C4P or C8P (or the GT option) then you have a different clock frequency. Look in your manual or phone your local OSI dealer to find out your clock frequency. The APPENDIX explains how to incorporate changes into the program for a different clock speed.

If the program still does not work, I can offer only the following general observations. If you run the program of Listing 2 and it causes your system to crash then the addresses in Line 670 are wrong, or the address in Line 540 is wrong. That is, the BASIC program can't find the printer jump table or it can't find a proper home for CHEEP PRINT. If the system crashes when you attempt to run CHEEP PRINT, then the program is not properly protected from BASIC (Lines 530-640) or the program can't find the printer jump table. If you hear a tone but no warbling, then again the problem is with the jump table. If you hear warbling, but get mish-mash, then the trouble is with either the tone timing (variables: LG; SH; SP; and WT) or the tone frequencies (variables: LO and HI). If the timing is off, your system clock is not 1MHZ. If the tone frequencies are off, you may be able to adjust them to the correct values by experimentation. The tone must sound clean and be clear of all clicks and pops.

Good luck! You will find that a printer gives a whole new dimension to your computing. If you have questions, I might be able to help. I'd be glad to reply if a question is accompanied by a stamped self-addressed envelope. Write to:

Tom Berger
10670 Hollywood Blvd.
Coon Rapids, MN. 55433

APPENDIX

All variables are named at the beginning of the BASIC loader program in Listing 2. The various standards are as follows:

BAUD RATE	BIT TIME(μS)	STOP BITS	STOP BITS(μS)	
110	9091	2	18182	
300	3333	1	3333	
MODE	MARK(1)	SPACE(0)	DIVISORS	ERRORS(%)
KANSAS CITY	2400HZ	1200HZ	20 41	2.4 0.1
ORIGINATE	2225HZ	2025HZ	22 24	0.4 1.1
ANSWER	1270HZ	1070HZ	39 46	0.8 0.1

The frequency of the tone generator is given by:

FREQUENCY = 49152/DIVISOR.

Below, T1 is the time of 1 bit. Time through the character loop:

T1 = 1284*LG + 5*SH - 1250

LG = INT((T1 + 1250)/1284)

SH = INT((T1 + 1250-1284*LG)/5)

The stop bit time is only approximately computed. If T is the full stop bit time then:

SP = INT((T + 1917)/1284).

If your clock speed is C MHZ then use INT(C*T1) and INT(C*T) in place of T1 and T in the above formulas.

Calculated parameters (1MHZ clock):

	110 BAUD	300 BAUD
LG	8	3
SH	14	146
SP	15	4

Calculated parameters (2 MHZ clock):

LG	15	6
SH	34	43
SP	30	7

Now the POKES of Listing 2 and their meanings:

POKE	NUMBER	MEANING
8960	ST	Address of the last memory page.
132,133	00,ST	BASIC end of memory.
8983	255	Low byte: Input/output jump table.
8984	ST-1	High byte: Same as above.
56832	3	Tone generator ON/OFF.
57089	HI, LO	Tone generator frequency.

```
510-

 510 ;
 520 ;TIME THROUGH LOOP2 TO RTS
 530         ;1284*LG + 5*SH - 1275
                 = T  (1 MHZ CLOCK)
 540 ;TIME THROUGH CHAR DECODE
 550         ;T + 25  (1 MHZ CLOCK)
 560 ;110 BAUD
 570         ;SH = $0E = 14
 580         ;LG = $08 =  8
 590         ;SP = $0F = 15
 600 ;300 BAUD
 610         ;SH = $92 = 146
 620         ;LG = $03 =   3
 630         ;SP = $04 =   4
 640 ;WAIT TIMES
 650         ;NCR WT = $14 =  20
 660         ;IBM WT = $FF = 255
 670         ;IBM SP = $28 =  40
 680 ;ANSWER MODE
 690         ;LO = $2E = 46
 700         ;HI = $27 = 39
 710 ;ORIGINATE MODE
 720         ;LO = $18 = 24
 730         ;HI = $16 = 22
 740 ;KANSAS CITY STANDARD
 750         ;LO = $29 = 41
 760         ;HI = $14 = 20
 770                 ;KC RUNS AT
                        300 BAUD
 780 .END
```

Listing 2

```
 100 REM *** A BASIC LOADER FOR
 110 REM *** "OSI CHEEP PRINT"
 120 REM *** T. R. BERGER 3/80
 130 REM *** VARIABLE NAMES
 140 REM DIVISOR LO TONE; LO
 150 REM DIVISOR HI TONE; HI
 160 REM STOP BIT TIME  ; SP
 170 REM CAR RETURN WAIT; WT
 180 REM DELAY LONG     ; LG
 190 REM DELAY SHORT    ; SH
 200 REM BAUD RATE      ; BR
 210 REM BYTE OF DATA   ; BY
 220 REM ADDR CHECK     ; CH
 230 REM ADDR DELAY HI  ; DH
 240 REM ADDR DELAY LO  ; DL
 250 REM ADDR LOOP2 HI  ; LH
 260 REM ADDR LOOP2 LO  ; LL
 270 REM ADDR START     ; ST
 280 REM 110 BAUD LONG  ; L1
 290 REM 110 BAUD SHORT ; S1
 300 REM 110 BAUD STOP  ; B1
 310 REM 300 BAUD LONG  ; L2
 320 REM 300 BAUD SHORT ; S2
 330 REM 300 BAUD STOP  ; B2
 340 REM KC TONE HI     ; H3
 350 REM KC TONE LO     ; L3
 360 REM ORG TONE HI    ; H4
 370 REM ORG TONE LO    ; L4
 380 REM ANS TONE HI    ; H5
 390 REM ANS TONE LO    ; L5
 400 REM MODE CHOICE    ; MO$
 410 REM STOP BIT CHANGE; SP$
 420 REM *** FIXED PARAMETERS
 430 REM 110 BAUD
 440 L1=8:S1=14:B1=15
 450 REM 300 BAUD
 460 L2=3:S2=146:B2=4
 470 REM KC MODE
 480 H3=20:L3=41
 490 REM ORG MODE
 500 H4=22:L4=24
 510 REM ANS MODE
 520 H5=39:L5=46
 530 REM *** PROTECT HI MEMORY
 540 ST=PEEK(8960)
 550 CH=(ST+1)*256
 OK

 560 REM IS PROGRAM LOADED?
 570 IF PEEK(CH)<>72 THEN 630
 580 IF PEEK(CH+1)<>120 THEN 630
 590 IF PEEK(CH+2)<>162 THEN 630
 600 ST=ST+1
 610 GOTO 670
 620 REM NO, THEN PROTECT
 630 POKE132,00:POKE133,ST
 640 POKE8960,ST-1
 650 REM *** DOS JUMP TABLE
 660 REM PRINTER SUBR ADDR
 670 POKE8983,255:POKE8984,ST-1
 680 REM *** START ADDRESS
 690 ST=ST*256
 700 REM *** CALC SUBR ADDR
 710 DH=INT((ST+46)/256)
 720 DL=(ST+46)-256*DH
 730 LH=INT((ST+50)/256)
 740 LL=(ST+50)-256*LH
 750 REM *** MAIN PROGRAM
 760 REM ** QUESTIONS FIRST
 770 PRINT "CHOOSE A MODE OF ";
 780 PRINT "OPERATION."
 790 PRINT "KANSAS CITY/ORIGINATE/";
 800 PRINT "ANSWER (K/O/A)";
```

```
810 INPUT MO$
820 IF MO$="K" THEN 850
830 IF MO$="O" THEN 850
840 IF MO$<>"A" THEN 770
850 PRINT "CHOOSE A BAUD RATE."
860 INPUT "(110/300)"; BR
870 IF BR=110 THEN 890
880 IF BR<>300 THEN 850
890 PRINT "CARRIAGE RETURN WAIT."
900 INPUT "(<256)"; WT
910 IF WT<0 OR WT>255 THEN 890
920 IF WT<>INT(WT) THEN 890
930 PRINT "CHANGE STOP BIT TIME?"
940 INPUT "(Y/N)"; SP$
950 IF SP$<>"Y"THEN 1020
960 PRINT "NEW TIME?"
970 INPUT "(<256)"; SP
980 IF SP<0 OR SP>255 THEN 930
990 IF SP<>INT(SP) THEN 930
1000 B1=SP: B2=SP
1010 REM ** LOAD CHEEP PRINT
1020 FOR I=0 TO 56
1030 READ BY
1040 POKE ST+I,BY
1050 NEXT I
OK
1060 REM ** BAUD RATE TIMING
1070 IF BR=110 THEN 1090
1080 LG=L2:SH=S2:SP=B2:GOTO 1110
1090 LG=L1:SH=S1:SP=B1
1100 REM ** TONE DIVISORS
1110 IF MO$="K" THEN 1140
1120 IF MO$="O" THEN 1150
1130 LO=L5:HI=H5:GOTO 1170
1140 LO=L3:HI=H3:GOTO 1170
1150 LO=L4:HI=H4
1160 REM ** INSERT PARAMETERS
1170 POKE ST+3,LO: POKE ST+13,LO
1180 POKE ST+17,HI
1190 POKE ST+40,WT
1200 POKE ST+30,SP
1210 POKE ST+47,LG:POKE ST+49,SH
1220 REM ** SUBROUTINE ADDRS.
1230 POKE ST+8,DL: POKE ST+9,DH
1240 POKE ST+22,DL:POKE ST+23,DH
1250 POKE ST+32,LL:POKE ST+33,LH
1260 POKE ST+42,LL:POKE ST+43,LH
1270 REM ** TURN ON TONE
1280 POKE 56832,3: POKE 57039,HI
1290 REM *** MACHINE PROGRAM
1300 DATA 72,120,162,0,142,1,223
1310 DATA 32,0,0,56,106,162,0
1320 DATA 144,2,162,0,142,1,223
1330 DATA 32,0,0,74,208,241,176
1340 DATA 239,162,0,32,0,0,104
1350 DATA 201,13,208,5,162,0,32
1360 DATA 0,0,88,96,162,0,160,0
1370 DATA 136,208,253,202,208
1380 DATA 250,96
1390 END
OK
```

Listing 3

```
100 REM *** TESTER FOR
110 REM *** "OSI CHEEP PRINT"
120 REM *** T.R. BERGER 3/80
130 REM ** GET START ADDR
140 ST=PEEK(8960)
150 ST=(ST+1)*256
160 REM ** DETERMINE MODE
170 INPUT "MODE (K/O/A)"; MO$
180 IF MO$="K" THEN 220
190 IF MO$="A" THEN 230
200 IF MO$="O" THEN 240
210 GOTO 170
220 LO=41:HI=20:GOTO 260
230 LO=46:HI=39:GOTO 260
240 LO=24:HI=22
250 REM ** SET I/O
260 DISK!"IO ,0A"
270 REM ** PRINT LOOP
280 REM VARY LO
290 FOR I=LO-2 TO LO+2
300 POKE ST+3,I: POKE ST+13,I
310 REM VARY HI
320 FOR J=HI-2 TO HI+2
330 REM KEEP HI ABOVE LO
340 IF I<=J THEN 510
350 POKE 57039,J
360 POKE ST+17,J
370 REM LET NEW TONE GO
380 FOR K=0 TO 1000
390 NEXT K
400 REM OUTPUT TWICE
410 FOR K=0 TO 1
420 PRINT "ABCDEFGHIJKLMNOPQRS";
430 PRINT "TUVWXYZ 0123456789"
440 PRINT "!#$%&'() =<>+-*/ .";
450 PRINT ":? "
460 PRINT "THIS IS A TEST."
470 PRINT "HI=";J;"LO=";I
480 PRINT : PRINT
490 NEXT K
500 NEXT J
510 NEXT I
520 PRINT "TEST COMPLETE."
530 REM ** RESET I/O
540 DISK!"IO ,02"
550 REM RESET TONES
560 POKE ST+3,LO: POKE ST+13,LO
570 POKE ST+17,HI
580 END
```

μ

$$ WIN $$

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.

E-Z DRAW and DUCK HUNT include character generation by Ron and Darrel Aldrich

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

PRINT USING for Applesoft

One of the minor but annoying problems with BASIC is the format of output. The program here permits user-defined formatting of the output for Applesoft, and can be easily modified for other flavors of BASIC.

Gary A. Morris
53 Fairfield Ave.
Norwalk, CT 06854

When I started using my APPLE for business programming, my biggest headache was formatting output for reports. I started out using various BASIC subroutines that barely performed the needed job and required a lot of overhead. Tired of using MID$, LEFT$, RIGHT$, and STR$, I decided to write a general-purpose print formatter using the USR function in Applesoft.

The routine is written entirely in assembly language, which is ideal for handling this sort of problem. It is used from BASIC by assigning the string variable ED$, the edit pattern showing how you want the output formatted. During a print statement when you use the USR function, the argument is evaluated and then printed in the format specified by the current value of ED$.

In the sample BASIC program (in figure 1) line 10 loads the machine language program into RAM at $300-$3A9. Then line 20 puts a "JMP $0300" at $000A, which is used by Applesoft to find the routine to be used. Lines 10 and 20 are only needed once at the beginning of a program. Line 30 assigns an edit pattern to the variable ED$. Line 40 is a sample print statement that uses the USR function. Line 50 assigns a value to *X* (that we want printed) rounded off to two decimal places, and line 60 does this. If you wanted to round to three places, the 100 would be changed to 1000 and the edit pattern would have to be changed to allow three digits after the decimal point. Note that any valid expression could be within the parenthesis of the USR function.

The routine works by taking the number that Applesoft would normally print out and filling up the edit pattern with those characters from right to left, skipping over decimal points, commas and special characters.

The output of the routine may be used wherever a BASIC PRINT statement can be used, such as printing to a disk file, to a printer, or just to the screen. It is especially desirable for creating fixed-length records in files.

The edit pattern can be fairly complex, as in figure 1, or it can be simply blanks. Using a blank pattern will cause the number to be right-justified within the number of blanks in the edit pattern. If the number is too large to fit in the edit pattern, the left-most digits will be truncated. Any special characters ($," ÷ %:*) in the edit pattern will be skipped, and the digits will fill in over blanks or numeric digits in the pattern.

The zeros are used in the edit pattern so that, if the number is small, there will always be zeros between the decimal point and the right-most column. If the number is too small to fill past the comma(s), then the extra commas will be replaced with blanks. When using an edit pattern with a decimal point, the argument for the function must be a whole number, or two decimal points will result. The edit pattern must be less than or equal to 16 characters in length. If it is greater, it will be cut off at 16.

The machine language program was written so that it can be located anywhere in addressable memory space. It is completely relocatable. That is, no changes are needed to run it at another address. It requires 169 ($A9) bytes of RAM. The program uses the same zero page locations that are assigned to Applesoft so that there are no conflicts. It also uses 752-767 ($2F0-$2FF) as a buffer to perform editing. This area is in the input buffer and is not used during printing (except when printin DOS commands).

How It Works

For those of you who would like to know how the program works, keep reading. Starting with the PRINT statement, the argument for the USR is evaluated and placed in the floating point accumulator by the BASIC interpreter. Then a JSR is made to $000A, where we have a JMP to the start of our subroutine.

At the beginning of the machine language subroutine, the Applesoft floating point accumulator is converted (lines 48-55) into a character string, in the format that Applesoft

would normally print it out. This is done by the Applesoft subroutines FPSTR1 and FPSTR2 (my names). These routines leave the resulting string at the bottom of the page used for the stack ($100).

The routine then searches (lines 57-75) the variable table to find ED$. When found, its value is moved (lines 77-83) to the buffer area ($2F0-$2FF).

After the program has all the necessary data, it starts to work. The length of the unformatted number is found (lines 85-90); and this number (an ASCII string right now) is then moved (lines 92-133) into the buffer, one character at a time, from right to left. The current character in the pattern is checked and, if it is a special character, it is skipped. Minus signs are carried over any digits in the pattern so that they will be on the left of the number. This process continues until we run out of characters to put in the pattern (or the pattern fills up), at which time any leftover commas are covered up (lines 135-145) with blanks.

Finally the program is ready to print out the result. Lines 147-152 print out all of the number, except the last digit (I'll explain this in a moment), using the output routine in Applesoft. This output routine does all of the necessary checking and conversion so that Applesoft's SPEED, INVERSE, and FLASH functions will work. The routine also sets the most significant bit of all outgoing ASCII characters.

The USR function must return a value to the BASIC program, which will be printed out by the BASIC interpreter, because we are in a PRINT statement. The last character of the buffer (which must be a digit) is taken and converted to an integer in the *Y* register and passed to Applesoft's integer to floating conversion routine (lines 154-161). This routine converts the integer (passed in the *A,Y* registers) into floating point in the floating point accumulator, which is just where we need it to pass back to BASIC.

Hardware Requirements

This program requires an APPLE II+, and APPLE II with an Applesoft card, or and APPLE II with a language card. It will work in any memory size system. A disk drive is not required.

If the appropriate changes are made to the JSR's and JMP in the machine language routine, the program can be used with RAM Applesoft (which loads in at $0800-2FFF). After keying in the code from figure 2, if you then key in the code from figure 3, it will run with RAM Applesoft instead.

Figure 1: **Sample Program**

```
]LIST

10 PRINT CHR$(4);"BLOAD EDIT.OBJECT
    CODE,A$300"
20 POKE 10,76:POKE 11,0:POKE 12,3
30 ED$="$   ,   0.00"
40 PRINT "SUB TOTAL...";USR(3495)
50 X=12345.67899
60 PRINT "NET TOTAL...";
    USR(INT(X*100+.5))
70 END

]RUN

SUB TOTAL...$     34.95
NET TOTAL...$12,345.68

]
```

Figure 2: **HEX DUMP**

```
0300- A5 52 48 20 34 ED 20 E7
0308- E3 68 85 52 A9 45 A2 C4
0310- 85 81 86 82 20 53 E0 A0
0318- 04 B1 9B 85 84 88 B1 9B
0320- 85 83 88 B1 9B C9 10 90
0328- 02 A9 10 85 D0 A8 88 B1
0330- 83 99 F0 02 88 10 F8 A0
0338- 00 B9 00 01 F0 03 C8 D0
0340- F8 A6 D0 88 B9 00 01 48
0348- 68 48 C9 2D D0 0E BD EF
0350- 02 C9 2D 90 16 CA D0 F0
0358- 68 18 90 35 BD EF 02 C9
0360- 20 F0 08 C9 3A F0 EE C9
0368- 30 90 EA 68 9D EF 02 CA
0370- F0 1F C0 00 D0 CD E8 18
0378- 90 10 BD EF 02 C9 24 F0
0380- 10 C9 2E B0 05 A9 20 9D
0388- EF 02 CA F0 04 E4 D0 90
0390- E9 A2 01 BD EF 02 20 5C
0398- DB E8 E4 D0 90 F5 BD EF
03A0- 02 49 30 A8 A9 00 4C F2
03A8- E2
```

***Figure 3:* MOD FOR RAM APPLESOFT**

```
>BLOAD PRINT USING.OBJECT,A$300
>CALL-151
*304:2B 25 20 DE 1B
*315:4C 18
*397:5F 13
*3A7:EB 1A
*3D0G
>BSAVE PRINT USING.OBJECT,A$300,L$A9
```

```
LINE# LOC CODE   LINE

0002 0000        ;*************************************
0003 0000        ;*                                   *
0004 0000        ;*     PRINT USING for APPLESOFT     *
0005 0000        ;*                                   *
0006 0000        ;*      (C) 1980 by Gary Morris      *
0007 0000        ;*       Norwalk, Connecticut        *
0008 0000        ;*                                   *
0009 0000        ;*     Commercial Rights Reserved    *
0010 0000        ;*                                   *
0011 0000        ;*ASM/65                  JAN 5 1980*
0012 0000        ;*************************************

0014 0000        ;The USR  function  requires a JMP  to
0015 0000        ;the start of  the routine. If 'START'
0016 0000        ;equals the address  where the routine
0017 0000        ;is loaded then the following will set
0018 0000        ;up the JMP:
0019 0000        ; 10 POKE 10,76
0020 0000        ; 20 POKE 11,START-INT(START/256)*256
0021 0000        ; 30 POKE 12,INT(START/256)

0023 0000        ;Variables:
0024 0000        AFLAG  =$52          ;flag for Applesoft
0025 0000        NAME   =$81          ;variable name
0026 0000        PNTR   =$83          ;pntr to edit pattern
0027 0000        VARBLE =$9B          ;pointer to variable
0028 0000        LENGTH =$D0          ;pattern length
0029 0000        BUFFER =$02F0        ;edit buffer
0030 0000        STRING =$0100        ;number put here as -
0031 0000                             ;a character string

0032 0000        ;ROM Applesoft subroutine addresses:
0033 0000        FPSTR1 =$ED34        ;floating to string -
0034 0000        FPSTR2 =$E3E7        ;conversion routines
0035 0000        COUT   =$DB5C        ;print an ascii char
0036 0000        INTFP  =$E2F2        ;INT to FP conversion
0037 0000        FIND   =$E053        ;find a variable
```

```
LINE# LOC CODE   LINE

0039 0000        ;RAM Applesoft subroutine addresses:
0040 0000        ;FPSTR1 =$252B       ;floating to string -
0041 0000        ;FPSTR2 =$1BDE       ;conversion routines
0042 0000        ;COUT   =$135F       ;print an ascii char
0043 0000        ;INTFP  =$1AEB       ;INT to FP conversion
0044 0000        ;FIND   =$184C       ;find a variable

0046 0000                *=$0300      ;Organize at $0300
0047 0300                             ;(relocatable)

0048 0300        ;First convert floating point accum to
0049 0300        ;an ASCII string...
0050 0300 A552   START  LDA AFLAG     ;save the flag
0051 0302 48           PHA
0052 0303 2034ED       JSR FPSTR1    ;convert floating
0053 0306 20E7E3       JSR FPSTR2    ;point to string
0054 0309 68           PLA
0055 030A 8552         STA AFLAG     ;restore flag

0057 030C        ;Now find the  variable (ED$) that has
0058 030C        ;the edit pattern.
0059 030C A945   SEARCH LDA #'E'     ;basic variable
0060 030E A2C4         LDX #$C4      ;name is ED$
0061 0310 8581         STA NAME
0062 0312 8682         STX NAME+1
0063 0314 2053E0       JSR FIND
0064 0317 A004         LDY #4
0065 0319 B19B         LDA (VARBLE),Y  ;get addr hi
0066 031B 8584         STA PNTR+1
0067 031D 88           DEY
0068 031E B19B         LDA (VARBLE),Y  ;get addr lo
0069 0320 8583         STA PNTR
0070 0322 88           DEY
0071 0323 B19B         LDA (VARBLE),Y  ;get length
0072 0325 C910         CMP #16
0073 0327 9002         BCC LENOK     ;maximum length
0074 0329 A910         LDA #16       ;allowed is 16!!!
0075 032B 85D0   LENOK STA LENGTH

0077 032D        ;Move the pattern to the buffer
0078 032D A8           TAY
0079 032E 88           DEY
0080 032F B183   LOOP2 LDA (PNTR),Y
0081 0331 99F002       STA BUFFER,Y
0082 0334 88           DEY
0083 0335 10F8         BPL LOOP2

0085 0337        ;Find the string end
0086 0337 A000         LDY #0
0087 0339 B90001 LOOP  LDA STRING,Y  ;get char
0088 033C F003         BEQ NEXT2
0089 033E C8           INY
0090 033F D0F8         BNE LOOP
```

```
LINE# LOC CODE   LINE

0092 0341         ;Move string to the buffer, from right
0093 0341         ;to  left,  filling  over  numbers but
0094 0341         ;skipping comma's and periods.
0095 0341         ;     If we come  to  a minus sign then
0096 0341         ;keep going left until the pattern has
0097 0341         ;a blank or a  comma,  then keep going
0098 0341         ;left  storing  blanks  in  the buffer
0099 0341         ;until it ends  or we come to a dollar
0100 0341         ;sign.

0102 0341 A6D0    NEXT2  LDX LENGTH   ;field width
0103 0343 88      EDLOOP DEY
0104 0344 B90001         LDA STRING,Y  ;get a character
0105 0347 48             PHA           ;save it
0106 0348 68      CHECK  PLA
0107 0349 48             PHA
0108 034A C92D           CMP #'-'      ;if a minus then
0109 034C D00E           BNE DIGIT     ;skip to a blank

0110 034E BDEF02 MINUS  LDA BUFFER-1,X
0111 0351 C92D           CMP #'-'
0112 0353 9016           BCC DROPIT
0113 0355 CA      SKIPIT DEX
0114 0356 D0F0           BNE CHECK
0115 0358 68             PLA
0116 0359 18             CLC
0117 035A 9035           BCC DONE
0118 035C BDEF02 DIGIT  LDA BUFFER-1,X
0119 035F C920           CMP #' '
0120 0361 F008           BEQ DROPIT
0121 0363 C93A           CMP #':'
0122 0365 F0EE           BEQ SKIPIT
0123 0367 C930           CMP #'0'
0124 0369 90EA           BCC SKIPIT
0125 036B 68      DROPIT PLA           ;get it back
0126 036C 9DEF02         STA BUFFER-1,X
0127 036F CA             DEX
0128 0370 F01F           BEQ DONE
0129 0372 C000           CPY #0       ;end of string?
0130 0374 D0CD           BNE EDLOOP
0131 0376 E8             INX
0132 0377 18             CLC
0133 0378 9010           BCC NEXT1

0135 037A BDEF02 BLANK  LDA BUFFER-1,X ;blank from
0136 037D C924           CMP #'$'      ;here to $
0137 037F F010           BEQ DONE
0138 0381 C92E           CMP #'.'
0139 0383 B005           BCS NEXT1
0140 0385 A920           LDA #' '
0141 0387 9DEF02         STA BUFFER-1,X
0142 038A CA      NEXT1  DEX
0143 038B F004           BEQ DONE
0144 038D E4D0           CPX LENGTH
0145 038F 90E9           BCC BLANK

0147 0391 A201    DONE   LDX #1
0148 0393 BDEF02 LOOP4   LDA BUFFER-1,X ;print the
0149 0396 205CDB         JSR COUT      ;output buffer
0150 0399 E8             INX           ;except last char
0151 039A E4D0           CPX LENGTH
0152 039C 90F5           BCC LOOP4

0154 039E         ;Take the last  char from  the buffer,
0155 039E         ;convert it to floating  and return it
0156 039E         ;to applesoft to be printed.
0157 039E BDEF02         LDA BUFFER-1,X
0158 03A1 4930           EOR #'0'
0159 03A3 A8             TAY           ;lo order byte
0160 03A4 A900           LDA #0        ;hi order byte
0161 03A6 4CF2E2         JMP INTFP     ;convert & return
0163 03A9                .END

ERRORS = 0000 <0000>
```

SYMBOL TABLE

SYMBOL	VALUE				
AFLAG	0052	BLANK	037A	BUFFER	02F0
CHECK	0348	COUT	DB5C	DIGIT	035C
DONE	0391	DROPIT	036B	EDLOOP	0343
FIND	E053	FPSTR1	ED34	FPSTR2	E3E7
INTFP	E2F2	LENGTH	00D0	LENOK	032B
LOOP	0339	LOOP2	032F	LOOP4	0393
MINUS	034E	NAME	0081	NEXT1	038A
NEXT2	0341	PNTR	0083	SEARCH	030C
SKIPIT	0355	START	0300	STRING	0100
VARBLE	009B				

END OF ASSEMBLY

μ

Skyles Electric Works

Presenting the Skyles MacroTeA

Text Editor

To help you write your program, MacroTeA includes a powerful text editor with **34 command functions:**

AUTO	Numbers lines automatically.
NUMBER	Automatically renumbers lines.
FORMAT	Outputs text file in easy-to-read columns.
COPY	Copies a line or group of lines to a new location.
MOVE	Moves a line or group of lines to a new location.
DELETE	Deletes a line or group of lines.
CLEAR	Clears the text file.
PRINT	Prints a line or group of lines to the PET screen.
PUT	Saves a line or group of lines of text on the tape (or disc).
GET	Loads a previously saved line or group of lines of text from the tape (or disc).
DUPLICATE	Copies text file modules from one tape recorder to the other. Stops on specific modules to allow changes before it is duplicated. This command makes an unlimited length program (text file) practical.
HARD	Prints out text file on printer.
ASSEMBLE	Assembles text file with or without a listing. Assembly may be specified for the object code (program) to be recorded or placed in RAM memory.
PASS	Does second pass of assembly. Another command that makes unlimited length text files (source code) practical.
RUN	Runs (executes) a previously assembled program.
SYMBOLS	Prints out the symbol table (label file).
SET	Gives complete control of the size and location of the text file (source file), label file (symbol table) and relocatable buffer.
DISK	Gives complete access to the eleven DOS commands; PUT GET NEW INITIALIZE DIRECTORY COPY DUPLICATE SCRATCH VALIDATE RENAME ERROR REPORT
EDIT	Offers unbelievably powerful search and replace capability. Many large computer assemblers lack this sophistication.
FIND	Searches text file for defined strings. Optionally prints them and counts them; i.e., this command counts number of characters in text file.
MANUSCRIPT	Eliminates line numbers on PRINT and HARD command. Makes MacroTea a true and powerful Text Editor.
BREAK	Breaks to the Monitor portion of MacroTea. A return to Text Editor without loss of text is possible.
USER	Improves or tailors MacroTea's Text Editor to user's needs; "Do-it-yourself" command.

Fast...Fast Assembler

Briefly, the pseudo-ops are:

- **BA** Commands the assembler to begin placing assembled code where indicated.
- **CE** Commands the assembler to continue assembly unless certain serious errors occur. All errors are printed out.
- **LS** Commands the assembler to start listing source (text file) from this point on.
- **LC** Commands the assembler to stop list source (text file) from this point in the program.
- **CT** Commands the assembler to continue that source program (text file) on tape.
- **OS** Commands the assembler to store the object code in memory.
- **OC** Commands the assembler to not store object code in memory.
- **MC** Commands the assembler to store object code at location different from the location in which it is assembling object code.
- **SE** Commands the assembler to store an external address.
- **DS** Commands the assembler to set aside a block of storage.
- **BY** Commands the assembler to store data.
- **SI** Commands the assembler to store an internal address.
- **DE** Commands the assembler to calculate an external label expression.
- **DI** Commands the assembler to calculate an internal label expression.
- **EN** Informs the assembler that this is the end of the program.
- **EJ** Commands the assembler to eject to top of page on printer copy.
- **SET** A directive not a pseudo-op, directs the assemblers to redefine the value of a label.

Macro Assembler

The macro pseudo-ops include:

MD	This is a macro beginning instruction definition.
ME	This is end of a macro instruction definition.
EC	Do not output macro-generated code in source listing.
ES	Do output macro-generated code in source listing.

Conditional Assembler

The conditional assembly pseudo-ops are:

IEQ	If the label expression is equal to zero, assemble this block of source code (text file).
INE	If the label expression is not equal to zero, assemble this block of source code (text file).
IPL	If the label expression is positive, assemble this block of source code.
IMI	If the label expression is negative, assemble this block of source code.
*******	This is the end of a block of source code.

Enhanced Monitor

. . . By having 16 powerful commands:

A	Automatic MacroTeA cold start from Monitor.
Z	Automatic MacroTeA warm start from Monitor.
F	Loads from tape object code program.
S	Saves to tape object code between locations specified.
D	Disassembles object code back to source listing.
M	Displays in memory object code starting at selected location. The normal PET screen edit may be used to change the object code.
R	Displays in register. Contents may be changed using PET screen edit capabilities.
H	Hunts memory for a particular group of object codes.
W	Allows you to walk through the program one step at a time.
B	Breakpoint to occur after specified number of passes past specified address.
Q	Start on specified address. Quit if STOP key or breakpoint occurs.
T	Transfers a program or part of a program from one memory area to another.
G	Go!! Runs machine language program starting at selected location.
X	Exits back to BASIC.
I	Display memory and decoded ASCII characters.
P	Pack (fill) memory with specified byte.

What are the other unique features of the MacroTeA?

- Labels up to 10 characters in length
- 50 different symbols to choose from for each character
- 10^{16} different labels possible
- Create executable object code in memory or store on tape
- Text editor may be used for composing letters, manuscripts, etc.
- Text may be loaded and stored from tape or disc
- Powerful two-cassette duplicator function
- String search capability
- Macros may be nested 32 deep
- 25 Assembler psuedo-ops
- 5 Conditional assembler psuedo-ops
- 40 Error codes to pinpoint problems
- 16 Error codes related to Macros
- Warm-start button
- Enhanced monitor with 16 commands

...a completely solid state firmware system ...all in ROM and RAM. No tapes to load. The system is available from the time you turn on your PET to the time you shut it off.

15 chips on a single high quality printed circuit board; interfaces with PET's parallel address and data bus or with Skyles Memory Adapter. A comprehensive 170 page manual is included

Truly, there is simply no other system of this magnitude at anywhere near this price. **$395.00 ***

**California residents: please add 6% or 6.5% sales tax as required*

VISA, MASTERCHARGE ORDERS CALL (800) 538-3083 (except California residents)
CALIFORNIA ORDERS PLEASE CALL (408) 257-9140

Skyles Electric Works
231 E South Whisman Road
Mountain View, CA 94041
(415) 965-1735

Define Your Own Function Key on PET

If there is a commonly used BASIC function which you would like to be able to call easily from the keyboard, then use the program provided here and see how easy it really is.

Werner Kolbe
Hardstr. 77
CH 5432 Neuenhof
Switzerland

The following small program "Function Key" enables you to define your own command using the key &. The definition of the command is stored in the second cassette buffer and is available as long as PET is not switched off, or until the command is defined new.

Using the Program

Load and run the program Function Key. Listng 1 is for the new ROMs and listing 2 is for the old ROMs. The program asks you for your command definition. You enter a BASIC program line for example:

?"THE TIME IS NOW " TI$

The line should not contain more than 77 characters, as usual for a BASIC program line. Then you press "RETURN" and your command is defined.

Now whenever you press &, followed by RETURN, your special command is executed. In our example, the time will be printed. It is also possible to use the command in a program. In the same way as PET executes a PRINT whenever it finds a ?, it will execute your command on the occurence of the &. The new command does not disturb any of the other functions of your PET, unless you own a floppy disk system. In this case the program must be changed slightly.

Program Description

Even though the program is short its operation is complex. The principle is as follows: Your BASIC code is inserted as line 0 in the program. It is then shifted into the second cassette buffer starting at dec. 895. A second line containing a SYS 870 is added. Then a loop is inserted into PET's CHAR GET routine to see if the character & occurs. If an & was found the registers and pointers are saved and then the processor is set to the "execute a line routine", where the code pointer is set to the place dec. 895. As the code ends with a SYS870, a jump back to the machine code is executed where the registers and pointers are restored before normal operation goes on.

I added a disassembly of the machine code (old ROMs) for those who want to understand the whole program in more detail (listing 3).

```
FUNCTION KEY
LISTING 1   NEW ROM

 5 PRINT"□"
 10 FORI=813TO890:READK:POKEI,K:NEXT
 20 OPEN1,0:PRINT"ENTER YOUR FUNTION !"
 40 PRINT"0:";
 45 GETA$:PRINTA$;:IFA$<>CHR$(13)THEN45
 50 PRINT"1SYS870":PRINT"GOTO60■";
 55 POKE158,3:FORI=623TO625:POKEI,13:NEXT:END
 60 FORI=895TO995:A=PEEK(I+134):POKEI,A:NEXT
 70 SYS813:NEW
 500 DATA169,76,133,112,169,58,133,113,169,3,133,114,96
 510 DATA230,119,208,2,230,120,32,118,0,201,38,240,3,76,118,0,140
 515 DATA254,3,142,253,3,165,119,133,1,165,120,133,2,186,134,0,169
 520 DATA127,133,119,169,3,133,120,76,247,198,166,0,154,165,1,133,119
 525 DATA165,2,133,120,172,254,3,174,253,3,76,58,3,0
READY.
```

FUNCTION KEY
LISTING 2 OLD ROM

```
 5 PRINT"□"
 10 FORI=813TO890:READK:POKEI,K:NEXT
 20 OPEN1,0:PRINT"ENTER YOUR FUNCTION !
 40 PRINT"0:";
 45 GETA$:PRINTA$;:IFA$<>CHR$(13)THEN45
 50 PRINT"1SYS870":PRINT"GOTO60■";
 55 POKE525,3:FORI=527TO529:POKEI,13:NEXT:END
 60 FORI=895TO995:A=PEEK(I+134):POKEI,A:NEXT
 70 SYS813:NEW
 500 DATA169,76,133,194,169,58,133,195,169,3,133,196,96
 510 DATA230,201,208,2,230,202,32,200,0,201,38,240,3,76,200,0,140
 515 DATA254,3,142,253,3,165,201,133,1,165,202,133,2,186,134,0,169
 520 DATA127,133,201,169,3,133,202,76,233,198,166,0,154,165,1,133,201
 525 DATA165,2,133,202,172,254,3,174,253,3,76,58,3,0
READY.
```

Listing 3

```
032D A9 4C          LDA =4C     Change the first 3
032F 85 C2          STA Z1        bytes of the CHR GET
0331 A9 3A          LDA =3A
0333 85 C3          STA Z2        routine to
0335 A9 03          LDA =03
0337 85 C4          STA Z3        JMP 033A
0339 60             RTS
033A E6 C9    J5    INC Z4      Get character
033C D0 02          BNE L1
033E E6 CA          INC Z5
0340 20 C8 00 L1    JSR J2
0343 C9 26          CMP =26     & ?
0345 F0 03          BEQ L3
0347 4C C8 00       JMP J2      no, return
034A 8C FE 03 L3    STY W1      save registers and
034D 8E FD 03       STX W2        stack pointer
0350 A5 C9          LDA Z4
0352 85 01          STA Z6
0354 A5 CA          LDA Z5
0356 85 02          STA Z7
0358 BA             TSX
0359 86 00          STX ZERO
035B A9 7F          LDA =7F
035D 85 C9          STA Z4
035F A9 03          LDA =03
0361 85 CA          STA Z5
0363 4C E9 C6       JMP J4      execute shifted pgm
0366 A6 00          LDX ZERO
0368 9A             TXS
0369 A5 01          LDA Z6      restore everything
036B 85 C9          STA Z4
036D A5 02          LDA Z7
036F 85 CA          STA Z5
0371 AC FE 03       LDY W1
0374 AE FD 03       LDX W2
0377 4C 3A 03       JMP J5      go on
037A 00             BRK
```

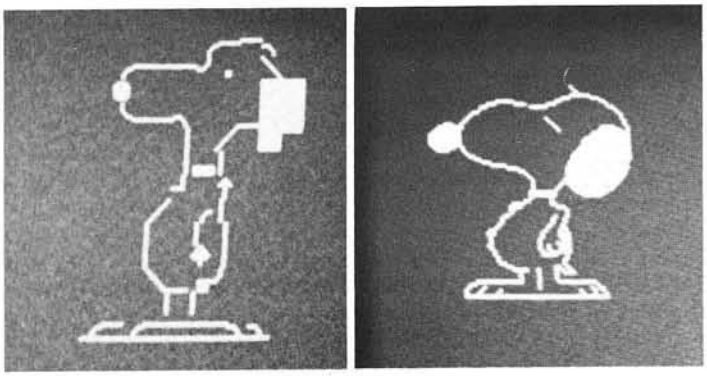
MICRO's FIRST ANNUAL
Limerick Contest Winner

The votes are in and we have a winner! Art Carpet of Canyon Country, California wins a year's free subscription to MICRO for his winning entry:

"There was a young hacker named
Drew
Who programmed all day and
night too.
By morning 'twas done
But he didn't type run
The poor little guy entered new."

Our congratulations to Mr. Carpet and our thanks to all who entered and/or voted in this, our first, Limerick Contest.

Judge this character generator for yourself



Regular PET Graphics HAL GRAPHICS

Compare the two Snoopys and you'll begin to see why the new HAL PCG 6500 is attracting so much attention. It's the programmable character generator for your PET that gives sharp, detailed graphics.

Important features include:

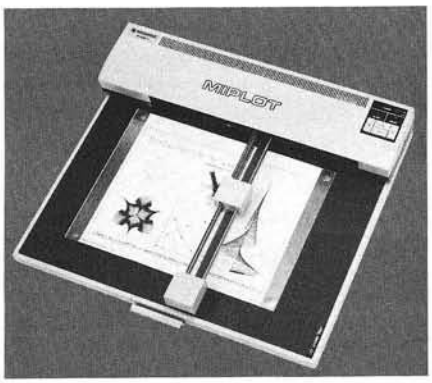
- Storage for 64 new characters
- Built-in CB2 sound amplifier that produces all tones in the human hearing range
- PCG Manual and demo program
- Interface to PET/CBM with 24 pin character generator ROMS. (If your PET uses 28 pin ROMS, Conversion Kits are available for $45. To simplify ordering, please indicate memory RAM #2114 or #6550.)

The HAL PCG 6500 is available exclusively at Systems Formulate ...

Only $200 (plus $5 shipping & handling)
Add $45 for 28 pin Conversion Kit

Exciting games are now available for the HAL PCG 6500 ... **Jupiter Lander, Night Driver, Super Slot** and more ... $19.95 each.

MIPLOT: the right plotter at the right price



MIPLOT by Watanabe Corporation

Designed for straight forward interface to any microcomputer that outputs the ASC11 code, MIPLOT can even be used by operators with no plotter experience.

- Incorporates pre-programmed "intelligent" functions required for producing graphs and drawings
- Solid and broken line types can be specified
- Built in character generator for letters, numbers and symbols
- Characters can be enlarged and rotated to four orientations
- Special printer mode outputs character data as-is
- Uses commonly available hard fiber-tip pens
- Maximum plot speed approximately 2 inches per second
- Built in self-test mode

Only $1,200
at Systems Formulate Corporation
(plus shipping & handling)

ADCOM Light Pen (with sound) ... Only $34.95

(plus $2 shipping & handling)

Unlike many light pens, the ADCOM interfaces with PET through the second cassette port rather than the parallel port. The result? Sound!

- Ready to plug in, no assembly necessary.
- Faster than many other light pens designed for PET.
- Built-in sensitivity control.
- Machine language and demo program included for ease of programming.

Expand with the finest in micro peripherals, systems and software from Systems Formulate.
Call today for more information or to place your phone order: (415) 326-9100
We honor Master Charge, Visa, check or money order.
(California residents add 6.5% tax)

SYSTEMS FORMULATE CORPORATION

39 Town & Country Village • Palo Alto, California 94301

An Improved Morse Code Receive Routine and Interface

Here are hardware and software enhancements to the article that appeared in Feb 1980 (21:19). The material contains information on interfacing to a 6522 VIA that is independent of the ham radio application.

Marvin L. DeJong
Dept. of Math & Physics
The School of the Ozarks
Point Lookout, MO 65726

In the February 1980 issue of MICRO (21:19) we described a program that used the AIM 65 keyboard to send Morse code and the AIM 65 20-character display to output received Morse code. The interface circuit that was used to interface the receiver to the transmitter for the purposes of receiving Morse code used an LM567 tone decoder. It appears from the feedback that I have received that the LM567 has such a narrow bandwidth that it is difficult to tune the received signal. What follows is a different approach to a receive routine, and a different interface. The program should be of interest to any computer owner who has a 6522 VIA in his system, not simply AIM 65 owners. With a few modifications, given in the Appendix, it can be used with the program described in the February issue of MICRO.

We begin by describing the new interface circuit that was used. Its function is to change the sine wave audio output from the receiver to pulses that have the shape of a square wave. These pulses are counted by the T2 counter/timer on the 6522. The interface circuit is shown in Figure 1. The LM386 acts as an audio amplifier, while the 555 is configured as a Schmitt trigger. The gain control on the receiver and the 10K potentiometer in the interface circuit are adjusted to give a comfortable listening level while the LED in the circuit in Figure 1 flashes only occasionally when no tone is present. That is, the noise level should be kept between the trigger thresholds of $\frac{1}{3}V_C$ and $\frac{2}{3}V_C$ on the 555 timer. When a tone from a transmitting station is in the passband of the reciever, the 555 should toggle back and forth at the same frequency as the tone input, producing a square wave input on PB6, and

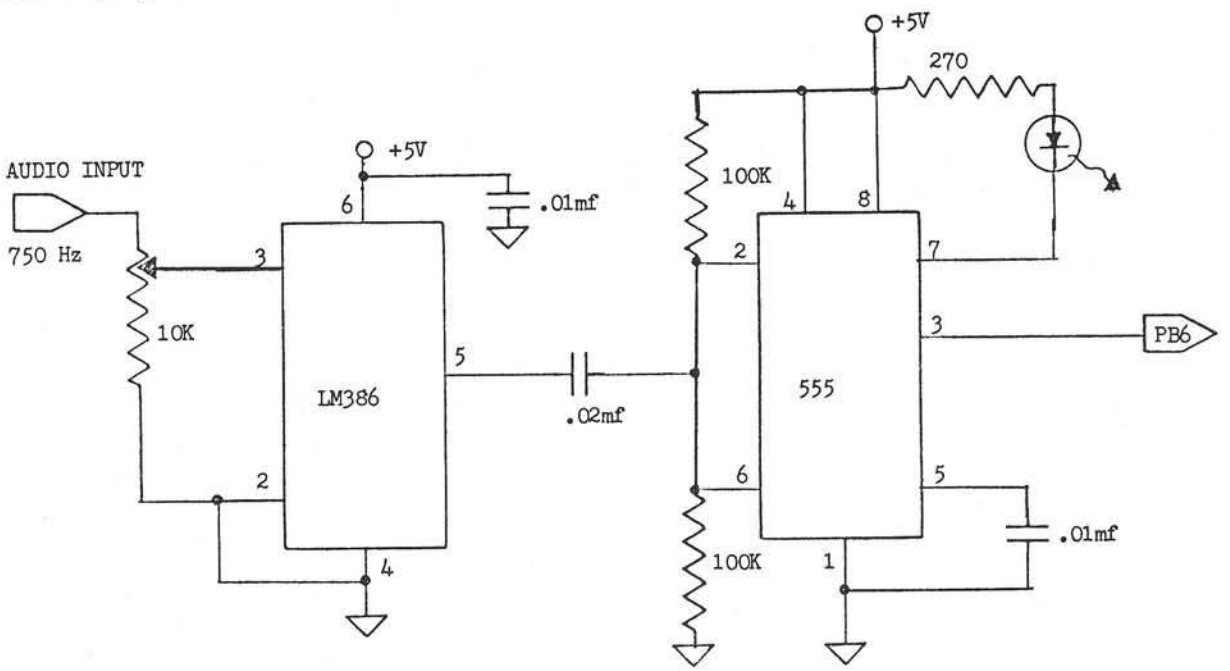


Figure 1: **Interface circuit between the receiver and PB6, the pulse counting pin.**

the LED will glow. These square wave pulses are counted by the T2 counter/timer. Counting noise pulses and static crashes (QRN) will produce various forms of useless and meaningless output, so keep the gain controls low enough to reject most of the noise. Best reception is obtained with a good sharp audio filter and just enough gain to detect the tone but not the noise.

The program is given in Listing 1. The part of the listing with comments is the basic code reception program that can be used with any microcomputer having a 6522 VIA and some kind of output device such as a teleprinter or a video monitor. Provided the subroutine jumps are modified accordingly, the commented program and subroutines are relocatable. The Morse-ASCII look-up table must also be loaded for the receive routine to work, and you must provide a suitable output routine that takes an ASCII character in the accumulator and outputs it. This is the JSR OUTCH instruction at $07BE. If you have an AIM 65, use the subroutines listed in the AIM 65 format, which may also be relocated.

The receive routine automatically sets the code speed and will copy up to about 70 words per minute. Speeds less than about 20 words per minute do not work very well because the locations that hold the various character length parameters overflow. The comments should make the general flow of the program clear. Basically, the T2 counter/timer counts pulses from the receiver. If a pulse is not counted, a space counter (SPACE) is incremented at a 750 Hz rate. Thus, the receive routine returns to subroutine COUNT once every 1/750 second. If a pulse has been counted, MARK is incremented, otherwise SPACE is incremented. Refer to the listing to see that in STATE1 the program is waiting for a mark of sufficient length (¾ of the dot length) to appear. When it does, the program makes the transition to STATE2, where it waits for the space counter to reach ½ the dot length. At this time it is assumed that the mark has been completed and the program jumps to STATE3 where it is decided whether the mark was a dot or a dash. The character register is modified accordingly, and the calibration

Listing 1: **Morse code receive routine.**

```
0F00 D8          START   CLD             Clear decimal mode for all additions.
0F01 A9 20               LDA $20         Set up T2 counter/timer to count pulses
0F03 8D 0B A0            STA ACR         by loading the ACR of the 6522 with $20.
0F06 85 0D               STA DOT         Also initialize dot counter to $20, and
0F08 85 0C               STA MARK        mark counter to $20.
0F0A 78                  SEI             Do not allow interrupts to bother this
0F0B 20 AC 0F            JSR CAL         routine.  Before starting to receive,
0F0E A9 01               LDA $01         calibrate the various registers.
0F10 85 0A               STA CHAR        Initialize the character register to $01.
0F12 20 80 0F            JSR TIMER       Jump to start space timer.
0F15 A9 00       STATE1  LDA $00         Clear mark and space counters.
0F17 85 0B       RPT     STA SPACE       Next we will jump to subroutine COUNT
0F19 85 0C               STA MARK        to increment the number of pulses from
0F1B 20 8B 0F    CNT     JSR COUNT       the receiver that have been detected.
0F1E A5 0C               LDA MARK        If the pulses have exceeded 3/4 those
0F20 C5 0F               CMP 3/4DOT      in DOT, a mark has been detected.  Then
0F22 B0 08               BCS STATE2      jump to STATE2 to wait for a space.
0F24 A5 0B               LDA SPACE       If space counter exceeds one dot, start
0F26 C5 0D               CMP DOT         over in STATE1.
0F28 B0 EB               BCS STATE1      Otherwise, continue counting marks and
0F2A 90 EF               BCC CNT         spaces.
0F2C A9 00       STATE2  LDA $00         Begin STATE2 by clearing space counter.
0F2E 85 0B               STA SPACE
0F30 20 8B 0F    MORE    JSR COUNT       Then count marks and spaces.  When SPACE
0F33 A5 0B               LDA SPACE       exceeds 1/2 dot length, then space has
0F35 C5 0E               CMP 1/2DOT      been detected.  Go to STATE3.  Otherwise
0F37 90 F7               BCC MORE        count some more.
0F39 06 0A       STATE3  ASL CHAR        A mark element has been detected, shift
0F3B A5 0C               LDA MARK        character register left.  If the mark
0F3D C5 12               CMP 2DOT        length exceeded 2 dots, then it was a dash.
0F3F 90 0C               BCC ARND        If a dot, do not put a one in CHAR.
0F41 E6 0A               INC CHAR        If a dash, add a one to the character register.
0F43 4A                  LSR A           Convert the dash length to a dot by multiplyin
0F44 4A                  LSR A           by 3/8.
0F45 85 0C               STA MARK
0F47 4A                  LSR A
0F48 18                  CLC
0F49 65 0C               ADC MARK
0F4B 85 0C               STA MARK
0F4D 20 AC 0F    ARND    JSR CAL           Jump to calibration routine to update
0F50 A9 00               LDA $00           automatic calibration of DOT.
0F52 85 0C               STA MARK          Clear mark counter.
0F54 20 8B 0F    LOAF    JSR COUNT         Start counting again.  If a new mark is
0F57 A5 0C               LDA MARK          detected, go to STATE2.  Otherwise wait
0F59 C5 0F               CMP 3/4DOT        for a character space.
0F5B B0 CF               BCS STATE2
0F5D A5 0B               LDA SPACE         Has space exceeded 2DOT?  That's a
0F5F C5 12               CMP 2DOT          character space so output the character.
0F61 90 F1               BCC LOAF **       Otherwise, return to count some more.
0F63 20 00 0E            JSR SPEED **      Subroutine speed measures the speed and
0F66 A9 01               LDA $01           outputs the speed and the new character.
0F68 85 0A               STA CHAR          Reset the character register.
0F6A 20 8B 0F    LOITER  JSR COUNT         Then wait for more pulses and spaces.
0F6D A5 0C               LDA MARK          Has a new mark been detected.  If so,
0F6F C5 0F               CMP 3/4DOT        jump to STATE2.
0F71 B0 B9               BCS STATE2
0F73 A5 0B               LDA SPACE         Does space exceed 5DOT.  If so, a
0F75 C5 13               CMP 5DOT          word space has been detected.
0F77 90 F1               BCC LOITER
0F79 20 B6 07            JSR OUTPUT        Output the word space.
0F7C A9 00               LDA $00           Clear accumulator.
0F7E F0 97               BEQ RPT           Back to STATE1.
                         ****************
                         SUBROUTINE TIMER
0F80 A9 A7       TIMER   LDA $A7           Load divid-by-eight timer for 167
0F82 8D 95 A4            STA TIME/8        counts, or 1336 microseconds = 1/750.
0F85 AD 08 A0            LDA T2CL          For 1000Hz tone use $7D rather than $A7.
0F88 85 18               STA STORE         Read T2 counter and store.
0F8A 60                  RTS               Then return.
                         ****************
                         SUBROUTINE COUNT
0F8B 58          COUNT   CLI               Allow an interrupt here (for author's
0F8C 78                  SEI               send routine.)
0F8D 2C 97 A4    WAIT    BIT TMFLAG        Has timer timed-out?
0F90 10 FB               BPL WAIT          No. Then wait.
0F92 38                  SEC               Yes. Then count the pulses that have
0F93 A5 18               LDA STORE         been detected by T2.
0F95 ED 08 A0            SBC T2CL
0F98 85 19               STA TEMP          Save pulse count here for a moment.
0F9A 20 80 0F            JSR TIMER         Start the timer again.
0F9D A5 19               LDA TEMP          Get the pulse count again.
0F9F F0 08               BEQ AHED          If no pulses, then increment space
0FA1 18                  CLC               counter.  If pulses, then add the
0FA2 65 0C               ADC MARK          number to MARK.
0FA4 85 0C               STA MARK
```

```
0FA6 18                 CLC             Skip around space increment instructions.
0FA7 90 02              BCC DETOUR
0FA9 E6 0B      AHED    INC SPACE       Increment space counter if no pulses
0FAB 60         DETOUR  RTS             detected, then back to the program.
**Change this to JSR OUTPUT (20 B6 07) if you do not have an AIM 65.
0FAC 46 0C      CAL     LSR MARK        Divide Mark by 2.
0FAE A5 0D              LDA DOT         Get previous DOT.
0FB0 46 0D              LSR DOT         Divide previous DOT by 2.
0FB2 18                 CLC
0FB3 65 0C              ADC MARK        DOT plus MARK/2.
0FB5 18                 CLC
0FB6 65 0D              ADC DOT         DOT plus MARK/2 plus DOT/2.
0FB8 4A                 LSR A           New DOT = 3/4 DOT plus 1/4 MARK
0FB9 C9 0F              CMP $0F         Is New DOT too short (noise problems)?
0FBB B0 02              BCS SKIP
0FBD A9 0F              LDA $0F         Yes, use $0F as minimum DOT length.
0FBF 85 0D      SKIP    STA DOT         We have a new DOT length.
0FC1 0A                 ASL A           Calculate 2DOT.
0FC2 85 12              STA 2DOT
0FC4 0A                 ASL A           Calculate 5DOT.
0FC5 18                 CLC
0FC6 65 0D              ADC DOT
0FC8 85 13              STA 5DOT
0FCA A5 0D              LDA DOT         Calculate 1/2DOT.
0FCC 4A                 LSR A
0FCD 85 0E              STA 1/2DOT
0FCF 4A                 LSR A           Calculate 3/4DOT.
0FD0 65 0E              ADC 1/2DOT
0FD2 85 0F              STA 3/4DOT
0FD4 60                 RTS
                        ****************
                        SUBROUTINE OUTPUT
07B6 A5 0A      OUTPUT  LDA CHAR        Get contents of character register.
07B8 09 80              ORA $80         Put a one in bit seven.
07BA A8                 TAY             Transfer this to the Y register and
07BB B9 00 04           LDA TAB,Y       look up the ASCII code in the look-up
07BE 20 72 06           JSR OUTCH       table.  Use your own subroutine to
07C1 20 60 06           JSR DISP        output the ASCII character to your
07C4 60                 RTS             output device (teleprinter or CRT).
                                        The routines here are for the AIM 65.
```

routine is called to update the dot length. The program then waits until a new mark is received, or if the space counter reaches twice the dot length then a character is printed. If a new mark is received the program jumps back to STATE2. When the space counter reaches five time the dot length, a word space is printed, and the program jumps back to STATE1.

AIM 65 users will want to use subroutine SPEED where it is called, while other microcomputer owners will use subroutine OUTPUT at this point. Subroutine SPEED converts the dot length to a code speed in words per minute and displays this number on the left-most digits on the AIM 65 display. Remember, this is only a crude approximation to the exact speed, unless your CW output tone on your receiver is *exactly* 750 Hz. The subroutine TIMER is set for a period corresponding to the period of the tone from the receiver. The byte given at $0F81 corresponds to 1333 microseconds, the period of a 750 Hz tone. This byte may be modified to be equal to the period of some other tone frequency if your receiver has a different tone frequency.

Appendix

To use this program with the send/receive program in the Feb. 1980 issue of MICRO, make the following changes in the listing given there, in addition to loading the program and subroutines listed here:

$0575	$60 instead of $40
$0655	$00 instead of $20
$0656	$0F instead of $08
$0F03	$0F05 $EA, $EA, $EA in Listing 1 in this article.
$07B6	Use the subroutine given in Listing 1.

I would really like to hear from someone who makes a comparison between the program and interface described here and the one described in the earlier article in MICRO. The program described here is a novel approach, intended to be more immune to noise, and has worked very well for me. But I would like to hear from you if you have any suggestions, improvements or criticisms.

AIM-65 output subroutines

```
0660 A2 LDX #13
0662 8A TXA
0663 48 PHA
0664 BD LDA A438,X
0667 09 ORA #80
0669 20 JSR EF7B
066C 68 PLA
066D AA TAX
066E CA DEX
066F 10 BPL 0662
0671 60 RTS
0672 8D STA A44C
0675 A2 LDX #03
0677 BD LDA A438,X
067A CA DEX
067B 9D STA A438,X
067E E8 INX
067F E8 INX
0680 E0 CPX #15
0682 90 BCC 0677
0684 60 RTS
0685 A2 LDX #10
0687 BD LDA A43A,X
068A E8 INX
068B 9D STA A43A,X
068E CA DEX
068F CA DEX
0690 10 BPL 0687
0692 A9 LDA #20
0694 8D STA A439
0697 20 JSR 0660
069A 60 RTS
069B A2 LDX #13
069D A9 LDA #20
069F 9D STA A438,X
06A2 CA DEX
06A3 10 BPL 069F
06A5 60 RTS
06A6 38 SEC
06A7 A5 LDA 22
06A9 E5 SBC 20
06AB C9 CMP #12
06AD B0 BCS 06BF
06AF 85 STA 24
06B1 38 SEC
06B2 A9 LDA #11
06B4 E5 SBC 24
06B6 AA TAX
06B7 A9 LDA #20
06B9 9D STA A43A,X
06BC 20 JSR 0660
06BF 60 RTS
```

Subroutine Speed

```
0E00  A5  LDA  0D
0E02  85  STA  10
0E04  A2  LDX  #00
0E06  A9  LDA  #84
0E08  85  STA  08
0E0A  A9  LDA  #03
0E0C  85  STA  09
0E0E  A5  LDA  08
0E10  E5  SBC  10
0E12  85  STA  08
0E14  A5  LDA  09
0E16  E9  SBC  #00
0E18  85  STA  09
0E1A  E8  INX
0E1B  B0  BCS  0E0E
0E1D  86  STX  08
0E1F  A9  LDA  #00
0E21  85  STA  10
0E23  F8  SED
0E24  A0  LDY  #08
0E26  06  ASL  08
0E28  A5  LDA  10
0E2A  65  ADC  10
0E2C  85  STA  10
0E2E  88  DEY
0E2F  D0  BNE  0E26
0E31  D8  CLD
0E32  A2  LDX  #01
0E34  A5  LDA  10
0E36  29  AND  #0F
0E38  18  CLC
0E39  69  ADC  #30
0E3B  9D  STA  A438,X
0E3E  A5  LDA  10
0E40  4A  LSR  .A
0E41  4A  LSR  .A
0E42  4A  LSR  .A
0E43  4A  LSR  .A
0E44  CA  DEX
0E45  30  BMI  0E49
0E47  10  BPL  0E38
0E49  20  JSR  07B6
0E4C  60  RTS
```

MORSE-ASCII Look-Up Table

```
<M>=0480 20 20 45 54
< > 0484 49 41 4E 40
< > 0488 53 55 52 57
< > 048C 44 48 47 4F
< > 0490 48 56 46 20
< > 0494 4C 20 50 4A
< > 0498 42 58 43 59
< > 049C 5A 51 20 20
< > 04A0 35 34 20 33
< > 04A4 20 20 20 32
< > 04A8 20 20 20 20
< > 04AC 20 20 20 31
< > 04B0 36 30 2F 20
< > 04B4 20 20 20 20
< > 04B8 37 20 20 20
< > 04BC 38 20 39 30
< > 04C0 20 20 20 20
< > 04C4 20 20 20 20
< > 04C8 20 20 20 20
< > 04CC 3F 20 20 20
< > 04D0 20 20 20 20
< > 04D4 20 2E 20 20
< > 04D8 20 20 20 20
< > 04DC 20 20 20 20
< > 04E0 20 20 20 20
< > 04E4 20 20 20 20
< > 04E8 20 20 20 20
< > 04EC 20 20 20 20
< > 04F0 20 20 20 20
```

μ

*Un*dedicating a Dedicated Microcomputer

Some valuable ideas are presented here toward developing a time-shared process controller.

David Borton
4 Kenworth Ave.
Troy, NY 12180

Introduction

Programs which use microcomputers for dedicated control often have two problems. One problem is lack of flexibility. Because the programs are either stored in EPROM or else they must be reassembled and reloaded into RAM, the control routines are difficult to change. A second problem is that the control routines can tie up the whole computer for only one dedicated application. While working on a project at Rensselaer Polytechnic Institute I have developed a dedicated controller that solves both of these problems.

Background

The project at RPI is the development of a solar energy collector. See Figure 1. The collector is a Fresnel reflecting concentrator which redirects solar power to a monotube boiler where cold water absorbs the solar energy and becomes hot steam. In order to keep the concentrated sunlight in the boiler, this type of collector follows the sun across the sky by means of two rotation axes. When the sun is out, this 2-axis tracking of the sun is straightforward and can be done with discrete logic without a microcomputer. Often, however, the sun is not out for variable portions of the day, and while it is behind clouds, it continues its motion across the sky. When the sun comes out again the collector is no longer aimed at the sun, and therefore the collector must have a large, high power, expensive motor and/or take a long time to reposition itself toward the sun. A microcomputer, however, can easily track the sun while it is behind clouds thus allowing the use of small, inexpensive, low power tracking motors. This article describes how a dedicated microcomputer is configured.

The System Hardware

The system could reside on a one chip micro but is currently made up of a KIM-1, a Memory Plus and an in-house interface board. The interface board has a 16 channel 8-bit A/D converter (see Joe William's article in MICRO 12:25) and digital I/O drivers and logic. The Memory Plus provides I/O lines, sockets for EPROM and RAM for data.

Another KIM and Memory Plus with an EPROM version of the MicroADE assembler are connected to a Teletype 43 for program development, testing and EPROM programing. Assembled programs are easily tested in RAM since the development system is almost identical to the control system. Source programs are stored on cassette tape for revisions.

The Implementation

The control program is made of modules of code in EPROM and

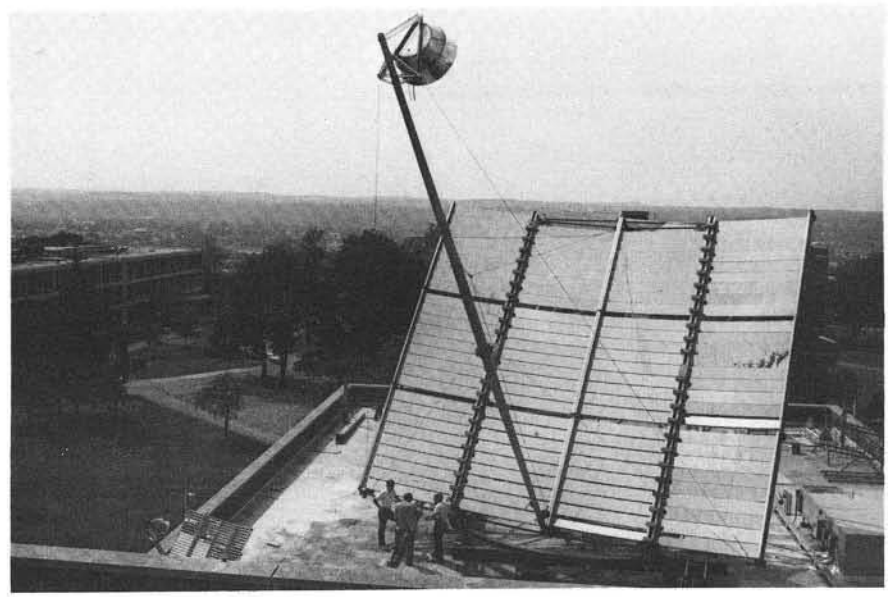


Figure 1

uses the non-maskable interrupt. It consists of two parts: the part that controls the position of the solar collector is the interrupt routine; the main loop part provides for other uses of the computer and interactively monitors the collector and/or the computer.

Interrupt Service

The non-maskable interrupt(NMI) is wired to the timer in one of the KIM 6530 chips. Every ¼ second, when the timer times out, the interrupt service routine is executed. See figure 2. This routine is made of modular sections, each of which performs some function. The first section of the NMI routine is a clock. This section fine tunes the ¼ second interrupt and keeps track of days as well as hours, minutes and seconds.

One of the NMI routine sections performs the A/D conversions as well as the I/O functions of sensing the sun and turning on and off motors and indicators. Another section actually makes the control decisions which the I/O section carries out.

All the sections are in EPROM and end with a 'Jump Indirect' instruction. This instruction solves the first problem mentioned above, that of inflexible dedicated controllers. The difficulty is overcome because the RAM zero page pointers used by the jump indirect instruction can point to any other function module to be included in the routine. Each section ends with a jump indirect pointing to the next section and so on, until the last section points to the code that restores the x, y and p registers and performs a 'Return from Interrupt'(RTI).

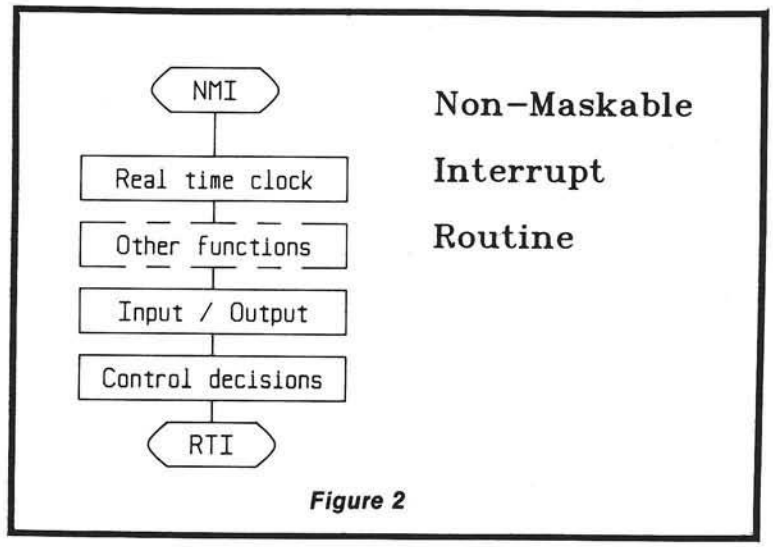


Figure 2

Main Loop

The main loop program (see figure 3.) is constructed in a way which solves the second problem, that of tying up the whole computer with just the control functions. The key to solving the problem is a keyboard monitor which allows other unrelated functions to be time-shared with the control functions. The keyboard monitor can select for display any of the A/D channels, the solar collector position, or the time of day. In addition, one of the unused analog inputs can be connected and monitored to use the microcomputer as a digital voltmeter.

The keyboard monitor can also select a jump to the KIM monitor. The KIM monitor can be used to enter or look at data or programs and even run other unrelated programs (including programs that use the IRQ, the maskable interrupt) while the dedicated controller is using the NMI and controlling the solar collector.

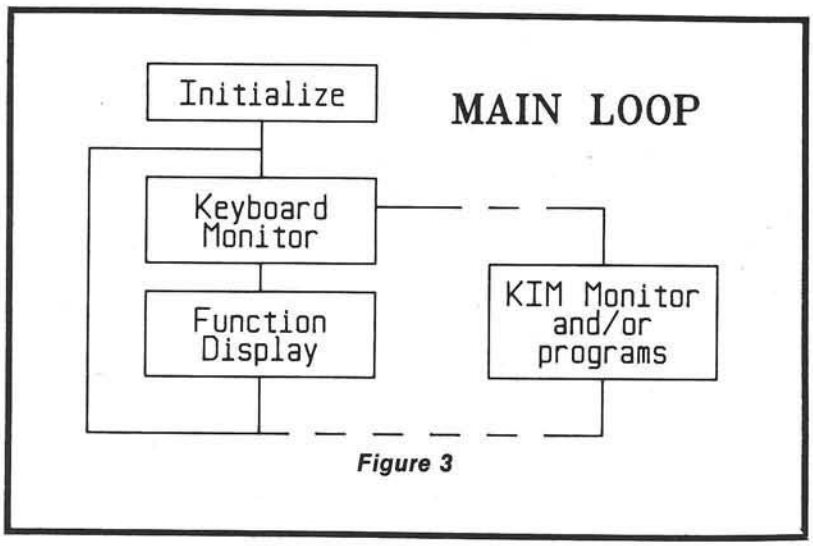


Figure 3

Cautions

There are minor hazards connected with having a flexible, time-shared dedicated controller. For example, a problem can occur when a program running on the system crashes, because, of course, pushing 'reset' stops the NMI clock timer. However, stopping the clock is not too serious a problem because with a warm restart of the control main loop, the clock will only lose a couple seconds, and it can be corrected at any time.

The warm restart is an important part of the initialization of the program. The original initialization sets up the NMI vector, the initial choices for the jump indirect pointers, the I/O ports and also starts the clock timer. By setting up the I/O and starting the timer at the end of the initialization a warm restart here will not change things like the time of the date or the jump indirect pointers that were correct before reset was pushed.

Conclusion

It is possible to program a dedicated controller which not only is flexible in what it does but, through time-sharing, can be used for other functions (and even act as a digital voltmeter) while it is performing as a dedicated controller.

μ

Attention

BARGAIN HUNTERS

Receive Hundreds of Classified Ads Like These Every Month

HARD DISK DRIVE Diablo Mod 31 1.2 MByte std. density. Includes power supp. and cable, rack mount slides, amd manual. Excellent condition. $450. Ca... 1601

IMPACT PRINTER 165 CPS Serial and parallel interfaces-Eight Selectable character sizes-Single and double width characters-uses standard plain paper - same mechanism as the integral data syst... 1 year old $589, Sta... St.,... 250...

HEATHKIT H-11/DEC LSI-11 system, 32K Byte storage, reader 1 punch, video terminal, complete software. Cost $4500 assembled, $3500 kit. Like new. Sell for $2250. 305-962-6677. 2058 Griffin Rd., Ft. Lauderdale, FL 33312.

FOR SALE: Interdata (Perkin-Elmer) 7/16 Mini with 32KB core, front panel, 50A PWR supply. Includes HS tape reader, interfaces for LP, 2 (TTY), and RS-232 (Full duplex, programmable). Includes manuals and much SW (Basic, Fortran, OS, etc.,). $800 - After 6 P... 2035...

COMPUTER AUTOMATION ALPHA 16; 16 k-word core memory,RTC, PF-R. Modified Mod. ASR-33 TTY Manuals, utilities, assemblers and many option boards - 16 bit I/O Driver, 16 bit I/O, Asynch modem contr. 64 bit output, 10 bit A/D - D/A. Fairly complete documentation. Up and running in Fortran. Not much more than TTY at $1000. **Herb Sauer, 303-494-8724.**

FOR SALE: Heath H9 video terminal, excellent condition, $175 or best offer. You ship. **[214] 962-4484**

WANTED: DIGITAL Group 32K memory board without memory chips and Phi deck controller board (kit, assembled or not working). W... T... 3...

PET COMPUTERS moving up to LSI-11. Pet business system priced to sell. PET 2001-16N Computer $800; 2040 Dual Floppy 340K (holds more data than 6 TRS-80 disks) $1,100. Digital cassettes (2) $60 each. System complete with Text Editor, disk sort, database software, real estate software and more $2,100. Call **PAUL** (313)971-8447

COMPUTER SHOPPER, the new buy, sell, and trade publication, is ready to help you with the latest information on personal, small business and large-system computers, accessories and software.

Each ad-packed issue is full of bargains you are looking for. Included are ads from individuals throughout the United States who are selling their good, pre-owned equipment just so they can trade-up to new equipment coming on the market.

But, COMPUTER SHOPPER'S bargains won't be yours unless you subscribe. This useful, money-saving publication can become your way to communicate with other buyers, sellers, and traders all over the nation.

Whether you are a hobbyist or a part-time user, COMPUTER SHOPPER will put you in touch with the nationwide computer marketplace in time for you to take advantage of bargain opportunities.

Have something to sell? A COMPUTER SHOPPER subscriber probably wants to buy it.

Looking for a part, component or even a complete system? A COMPUTER SHOPPER subscriber probably wants to sell it.

COMPUTER SHOPPER is THE marketplace for anything in computers and is read by thousands of people who are ready to buy.

COMPUTER SHOPPER offers a unique format in which classified ads are categorized for fast location of specific items. Combining this with low individual ad rates — 10 cents a word — makes it the ideal place for buyers and sellers to communicate. And, its mix of individual, dealer, and manufacturer ads enable subscribers to find what they want at the best price possible.

COMPUTER SHOPPER will work for you in other ways, too. If you are just thinking about getting into computers, it can help you learn product availability and prices before you make a decision. And, through the timely ads, COMPUTER SHOPPER will keep you abreast of changes in the market which could create bargain opportunities for you.

BUT COMPUTER SHOPPER cannot work for you unless you subscribe.

Want to look us over first? We'll give you your first issue FREE and then bill you for the next 12. If you are not convinced COMPUTER SHOPPER suits your needs, just write "cancel" on the invoice and return it.

And, to let COMPUTER SHOPPER start working for you right now, with a paid subscription we'll also give you a FREE classified ad to sell your pre-owned equipment or to find equipment you want.

If you don't need to use the free classified ad now, use it anytime during your subscription.

Subscription: $10/year, 12 issues plus your first free one. Bank cards accepted. Money back guarantee.

The Nationwide Marketplace for Computer Equipment

COMPUTER SHOPPER

P.O. Box F22 ● Titusville, Florida 32780
Telephone 305-269-3211

MasterCharge or VISA orders only, call TOLL FREE 800-327-9920.

A "Stop-On-Address" Routine for KIM

This very useful subroutine allows the user to debug more efficiently through the use of a trap, which will stop on a specified address.

R. MacDonald
45 Charles Tupper Drive
Westhill, Ontario
M1C 2A8

In the midst of debugging machine code for the KIM-1, it is this author's common practice to insert jumps to the monitor or breaks at strategic points in the program, and then start execution from the beginning. This is usually quite useful for tracing through a program to see what has happened to the output that was supposed to appear. There are two main problems with this technique however:

1. You have to write down the contents of where you are placing the breakpoint and then return the contents later.
2. The method is not usable for investigating a ROM or PROM.

A Solution

One means of getting around these difficulties is to use a tracer program. This can be accomplished quite easily by using a hardware timer, operating in the interrupt mode (eg. ref. 1), to cause an interrupt during every instruction until a specified end address is reached. For KIM this is simple, since one need only duplicate the monitor's single-step routine (to handle the interrupts) with the addition of setting up the timer to interrupt during every instruction.

The Program

The program (listing 1) uses the KIM monitor page zero locations to save all registers. This allows you to single step from the selected address once the routine has operated. Register contents are inspected in the same manner as single step operation. As usual, pushing PC returns the next address to be executed.

To use the program do the following:

1. Set the vector for single step operation.
2. Set up PCL, PCH with the FIRST ADDRESS to be executed (this is not necessary following single step operation).
3. Place the END ADDRESS at ENDL ($1780) and ENDH ($1781).
4. Execute from START ($1782).

The program stops on the selected address (before executing it).

```
0010:                   STOP ON ADDRESS FOR KIM
0020:                   RICK MACDONALD
0030:
0040: 17DC              PCL     *       $00EF   KIM MONITOR PAGE ZERO
0050: 17DC              PCH     *       $00F0   EQUATES
0060: 17DC              PREG    *       $00F1
0070: 17DC              SPUSER  *       $00F2
0080: 17DC              ACC     *       $00F3
0090: 17DC              YREG    *       $00F4
0100: 17DC              XREG    *       $00F5
0110: 17DC              POINTL  *       $00FA
0120: 17DC              POINTH  *       $00FB
0130:
0140: 17DC              TIMCLR  *       $1704   STOP TIMER ADDRESS
0150: 17DC              TIMSET  *       $170C   START TIMER ADDRESS
0160:
0170: 1780                      ORG     $1780
0180:
0190: 1780 00           ENDL    =       $00     STORAGE FOR END ADDRESS
0200: 1781 00           ENDH    =       $00
0210:
0220: 1782 A9 A2        START   LDAIM   IRQ     SETUP IRQ VECTOR
0230: 1784 8D FE 17             STA     $17FE
0240: 1787 A9 17                LDAIM   IRQ     / HIGH ADDRESS
0250: 1789 8D FF 17             STA     $17FF
0260:
0270: 178C A6 F2        AGAIN   LDXZ    SPUSER  LOAD STACK POINTER
0280: 178E 9A                   TXS
0290: 178F A6 F5                LDXZ    XREG    LOAD X
0300: 1791 A4 F4                LDYZ    YREG    LOAD Y
0310: 1793 58                   CLI             ENABLE IRQ
0320: 1794 A9 11                LDAIM   $11     SET TIMER TO INTERRUPT
0330: 1796 8D 0C 17             STA     TIMSET  IN 17. CYCLES
0340: 1799 A5 F1                LDAZ    PREG    LOAD STATUS
```

For executing through a loop a number of times, pick an address in the loop to stop on and initialize LOOP (at $17DB) to the number of times through the loop required.

Notes

1. The single step switch must be off!
2. PB7 must be connected to IRQ.
3. The program fits quite nicely in the KIM ROM scratchpad area but is fully relocatable.
4. The routine is easily modified to stop on any register or memory location equal to a given value. Just replace "conditions 1 and 2" in the program with the appropriate test logic.

References

1. Larry Fish, *Kilobaud* #8, August 1977, p. 112. "Troubleshoot Your Software" A trace program for a TIM based 6502 system.

2. KIM-1 User Manual, MOS Technology.


```
0350: 179E 48              PHA          PUT IN STACK
0360: 179C A5 F3           LDAZ  ACC    GET ACCUMULATOR
0370: 179E 28              PLP          GET STATUS
0380: 179F 6C EF 00        JMI   PCL    EXECUTE INSTRUCTION
0390:
0400: 17A2 85 F3      IRQ  STAZ  ACC    THIS IS THE SAME AS
0410: 17A4 68              PLA          THE KIM MONITOR AT $1C00
0420: 17A5 85 F1           STAZ  PREG
0430: 17A7 68              PLA
0440: 17A8 85 EF           STAZ  PCL
0450: 17AA 85 FA           STAZ  POINTL
0460: 17AC 68              PLA
0470: 17AD 85 F0           STAZ  PCH
0480: 17AF 85 FB           STAZ  POINTH
0490: 17B1 84 F4           STYZ  YREG
0500: 17B3 86 F5           STXZ  XREG
0510: 17B5 BA              TSX
0520: 17B6 86 F2           STXZ  SPUSER
0530: 17B8 8D 04 17        STA   TIMCLR STOP TIMER
0540: 17BE AD 80 17        LDA   ENDL   CONDITION 1
0550: 17BE EA              NOP
0560: 17BF C5 EF           CMPZ  PCL
0570: 17C1 D0 C5           BNE   AGAIN
0580: 17C3 AD 81 17        LDA   ENDH   CONDITION 2
0590: 17C6 EA              NOP
0600: 17C7 C5 F0           CMPZ  PCH
0610: 17C9 D0 C1           BNE   AGAIN
0620: 17CB AD DB 17        LDA   LOOP
0630: 17CE D0 03           BNE   SKIP
0640: 17D0 4C 16 1C        JMP   $1C16  TO KIM IF ADDRESS AND NO LOOP
0650:
0660: 17D3 CE DB 17   SKIP DEC   LOOP
0670: 17D6 D0 B4           BNE   AGAIN
0680: 17D8 4C 16 1C        JMP   $1C16
0690:
0700: 17DB 00         LOOP =     $00    00 OR 01 TO EXECUTE ONCE
0710:                      02 OR MORE TO EXECUTE THAT NUMBER OF TIMES
0720:
1D=                        μ
```

Tiny Pilot Complemental (Co-Pilot)

A modification to Tiny Pilot which improves the "Match" statement.

Robert Schultz
231 West Duvall Street
Philadelphia, PA 19144

As soon as I installed the first 4K additional memory in my KIM, I figured that the time was right to try my hand at programming in something other than assembly language. The Tiny Pilot interpreter, written by Nicholas Vrtis, that appeared in MICRO, September 1979 (16:41), looked good. It is written for the SYM, but Bob Applegate's suggestions in MICRO, Feburary 1980 (21:41), helped me get it running on my system.

It is a fun language! But, to my disappointment, I discovered that the "Match" statement would only match the first word input into the accept field. For example, I was trying to get the program to match if the word "wrong" were typed by the person using the program. If he or she typed "wrong" it would match, but if "you are wrong" were typed, it wouldn't match.

I made a modification to the Tiny Pilot interpreter to remedy this shortcoming. This modification fits in the space in KIM's upper RAM right behind Applegate's I/O routines.

Here's how it works. The interpreter enters the old "Match" routine. If a match is made, fine, the interpreter goes on to the next routine. If no match is found, the modifications prevent the old "Match" routine from giving up that easily.

The modification routine looks for the first ASCII "space" in the accept

```
0010:                   TINY PILOT MODIFICATION
0020:                   ROBERT SCHULTZ
0030:
0040:                   "CO-PILOT"
0050:
0060: 17D5              FLG     *       $0002
0070: 17D5              NORAD   *       $009A
0080: 17D5              OUTCH   *       $1EA0   KIM OUTPUT CHARACTER
0090: 17D5              GETCH   *       $1E5A   KIM INPUT CHARACTER
0100: 17D5              CRLF    *       $1E2F   KIM CARRIAGE RETURN/LINEFEED
0110: 0353                      ORG     $0353   TIE IN TO EXISTING PROGRAM
0120:
0130: 0353 4C BC 17             JMP     $17BC   CHANGE FUNCTION OF JUMP
0140: 0356 4C A6 17             JMP     $17A6   JMP TO KIM UPPER RAM ON "A"
0150:
0160: 035C                      ORG     $035C
0170: 035C A6 9A                LDX     NORAD   CHANGE LOADING INST TO NORAD
0180:
0190: 036C                      ORG     $036C
0200: 036C D0 E5                BNE     $0353
0210:
0220: 03A3                      ORG     $03A3
0230: 03A3 4C B5 17             JMP     $17B5   JMP TO BEGINNING OF NEW LOOP
0240:
0250: 1780                      ORG     $1780   REGISTER PRESERVER
0260:
0270: 1780 00           XSAVE   =       $00
0280: 1781 00           YSAVE   =       $00
0290:
0300:                   SYM OUTCHR = SAVOUT
0310:
0320: 1782 8C 81 17     SAVOUT  STY     YSAVE   SAVE Y
0330: 1785 20 A0 1E             JSR     OUTCH   PRINT "A"
0340: 1788 AC 81 17             LDY     YSAVE   RESTORE Y
0350: 178B 60                   RTS             RETURN
0360:
0370:                   SYM INCHR = SAVIN
0380:
0390: 178C 8C 81 17     SAVIN   STY     YSAVE   SAVE Y
0400: 178F 20 5A 1E             JSR     GETCH   GET ASCII CHAR
0410: 1792 AC 81 17             LDY     YSAVE   RESTORE Y
0420: 1795 60                   RTS             RETURN
0430:
0440:                   SYM CRLF = SAVCR
0450:
0460: 1796 8C 81 17     SAVCR   STY     YSAVE   SAVE Y
0470: 1799 8E 80 17             STX     XSAVE   SAVE X
```

field. When it finds one, it makes the next character in the accept field into the first character of a new search.

If the "Match" routine doesn't find a match during this search, it looks for the next ASCII "space", and the search begins anew. When the modification routine discovers that the next possible accept field start position is the last position in the accept field, it calls it quits, declares no match, and goes on to the the next routine in the interpreter.

The following is a listing of my modification routine. The address locations that begin with "03" replace listings in Mr. Vrtis' program. They jump Tiny Pilot to the new routine in KIM's upper RAM. This new routine goes from 17A6 to 17D4. I've also supplied listings of the I/O routines suggested by Bob Applegate. They go from 1780 to 17A5.

I'd like to thank Mr. Vrtis for providing us with such a fun language in Tiny Pilot. I hope soon to be able to supply intriguing programs in Tiny Pilot.

```
0480: 179C 20 2F 1E          JSR   CRLF    OUTPUT CRLF
0490: 179F AC 81 17          LDY   YSAVE   RESTORE Y
0500: 17A2 AE 80 17          LDX   XSAVE   RESTORE X
0510: 17A5 60                RTS           RETURN
0520:
0530:                    INITIALIZE FOR NEW LOOP
0540:
0550: 17A6 C9 4D         MTEST  CMPIM $4D    TEST "M"
0560: 17A8 D0 07               BNE   NOTM
0570: 17AA A2 27               LDXIM $27     LOAD X WITH BUFFER OFFSET
0580: 17AC 86 9A               STX   NORAD
0590: 17AE 4C 5A 03            JMP   $035A   JMP TO BEGINNING OF LOOP
0600: 17B1 4C A9 03      NOTM  JMP   $03A9   JMP TO NEXT FUNCTION
0610: 17B4 EA                  NOP
0620:
0630:                    DETERMINE IF ALL WORDS IN BUFFER HAVE BEEN
0640:                    EXAMINED.  IF NOT, RESET BUFFER POINTER TO
0650:                    NEXT WORD
0660:
0670: 17B5 A6 9A         NLOOP  LDX   NORAD  LOAD BUFFER POINTER
0680: 17B7 CA            NNLOOP DEX          NEXT BUFFER POSITION
0690: 17B8 D0 07               BNE   BLOOP
0700: 17BA A2 4E               LDXIM $4E     NO, SO LOAD N IN X
0710: 17BC 86 02               STX   FLG     STORE IN FLG
0720: 17BE 4C 79 02            JMP   $0279   NEXT ROUTINE
0730:
0740: 17C1 B5 03         BLOOP  LDAZX $03    LOAD A FROM BUFFER
0750: 17C3 C9 20               CMPIM $20     TEST SPACE
0760: 17C5 F0 02               BEQ   CLOOP   BRANCH IF SPACE
0770: 17C7 D0 EE               BNE   NNLOOP
0780: 17C9 CA            CLOOP  DEX          NEXT CHARACTER
0790: 17CA A0 27               LDYIM $27     FIRST BUFFER POSITION
0800: 17CC 84 8D               STY   $008D   STORE INDEX POINTER
0810: 17CE A0 02               LDYIM $02     LOAD Y WITH CORRECT CONTENT
0820: 17D0 86 9A               STX   NORAD   PUT NEW BUFFER START INTO NORAD
0830: 17D2 4C 5A 03            JMP   $035A   JUMP TO BEGINNING OF NEW LOOP
0840:
ID=
```

μ

MICRO Club Circuit

Here is another listing of club information. The response to this feature has been great. Keep them coming! A club-special reprint offer is made below.

MICRO ATTENDS THE APPLE FAIR

The APPLE FAIR was held on August 16th at NYU Tisch Hall. Eight sponsoring clubs gathered together to present seminars and exhibitions for over 800 people in attendance. Each club contributed a disk to exhibit from its own library. This proved to be a very popular feature. Many of these clubs had never worked together before and were pleased to find their combined efforts produced an exciting event.

Seminars were held all day and were well attended to the very last. MICRO's Software Consultant, Ford Cavallari, gave a presentation on "The APPLE in Education." Fair Organizer, Bruce Brewster, commented that the idea of a unified effort for such an event, which brings many clubs together, has great merit. He expressed his hopes for more events similar to this in the future.

More than 10 dealers were present, along with MICRO. Our observations are that this was one of the best fairs ever run and was well worth the journey. We hope that others initiate similar events.

The Sponsoring Clubs for this APPLE Fair were:

BIG APPLE USERS' GROUP
APPLE POWER
N.J. APPLE USERS' GROUP
PHILADELPHIA AREA USERS' GROUP
APPLESHARE
APPLELIST
NEW ENGLAND APPLE TREE
and
WASHINGTON APPLE PI

MICRO distributed reprints of Professor William F. Luebbert's "What's Where in the APPLE." MICRO now offers them to any officially registered club. Simply have your secretary notify us and we will send you a quantity of these reprints.

If your club is not registered, why isn't it? Send for our registration form and indicate the amount of reprints you would like. We have on hand only a limited amount of this fine article, so be sure to have your secretary respond quickly!

OMEGA
OSI Microcomputer Enthusiasts Group — Australia

Because these members are spread all over Australia this club does not have regular meeting times. However, small groups of members in each city will often meet at each other's homes on an informal basis. OMEGA functions as a Hardware and Software Exchange, and publishes a bimonthly Newsletter. Membership is currently at 50, with 3-4 new members each month. For more information, contact:

Geoff Cohen
72 Spofforth Street
Holt
A.C.T., 2615
Australia

The Ann Arbor Apple

Meets every third Thursday of the month. Starting time is 7:00 p.m. at NCE/Compumart (1250 N. Main Street, Ann Arbor). Formed in January of 1980, this club now has a group of 80 members. Contact:

The Ann Arbor Apple
P.O.Box M-1047
Ann Arbor, MI 48106

"Formed so that Apple users could see some of the Hardware and Software available at meetings. Our main asset is our monthly, 22 page Newsletter. Dues are $12.00 per year and include free copying of club disks or discount by mail."

OKC Atari User Group

Meets once a month on announced dates (at previous meeting) at 7:30 p.m. at High Technology Computer Store (1611 N.W. 23rd Street). Terry Barker is club President for about 10 members, so far. For further information, contact:

Terry Barker
First National Bank
Box 25189
Oklahoma City, OK 73125

"To further understanding of the Atari."

Apple Sauce of Omaha

Meets on the last Wednesday of each month at 7:30 p.m. at the Engineering Building on the University of Nebraska Campus in Omaha. This group was formed in January of 1980 and has grown to over 40 members. Contact:

John Anderson
3701 McKinley 4B
Omaha, Nebraska 68112

"Our goal is to provide members with information about the Apple Computer, available software, and a forum to share member's knowledge of the Apple."

Updates—Updates—Updates

Santa Barbara Apple User Group

Address change for this club:

2007 State Street
Santa Barbara, CA 93105

Meets on the 1st Wednesday of the month at 7:30 p.m. at The Computer Shop (2007 State Street, Santa Barbara, CA). Correspondence should be addressed to:

Secretary
2031 De La Vina
Santa Barbara, CA 93105

New England Apple Tree

The zip code for the post office box as listed in MICRO 25:69 should be 01801. The Club's new president is Lori Steinmetz.

Speed up your PET programming with The BASIC Programmer's Toolkit,™ now only $39.95.

Don't waste valuable programming time if there's an easier way to go. Here it is: The BASIC Programmer's Toolkit, created by the Palo Alto ICs, a division of Nestar. The Toolkit is a set of super programming aids designed to enhance the writing, debugging and polishing of BASIC programs for the PET.

Increase your PET's IQ for $39.95.™

The BASIC Programmer's Toolkit has two kilobytes of ROM firmware on a single chip. This extra ROM store lets you avoid loading tapes or giving up valuable RAM storage. It plugs into a socket inside your PET system, or is mounted on a circuit board attached on the side of your PET, depending on which PET model you own.

There are basically two versions of PET. To determine which Toolkit you need, just turn on your PET. If you see ***COMMODORE BASIC*** your PET uses the TK-80P Toolkit. If you see ###COMMODORE BASIC###, your PET uses the TK-160 Toolkit. Other versions of the BASIC Programmer's Toolkit are available for PET systems that have been upgraded with additional memory.

How Toolkit makes your programming easier:

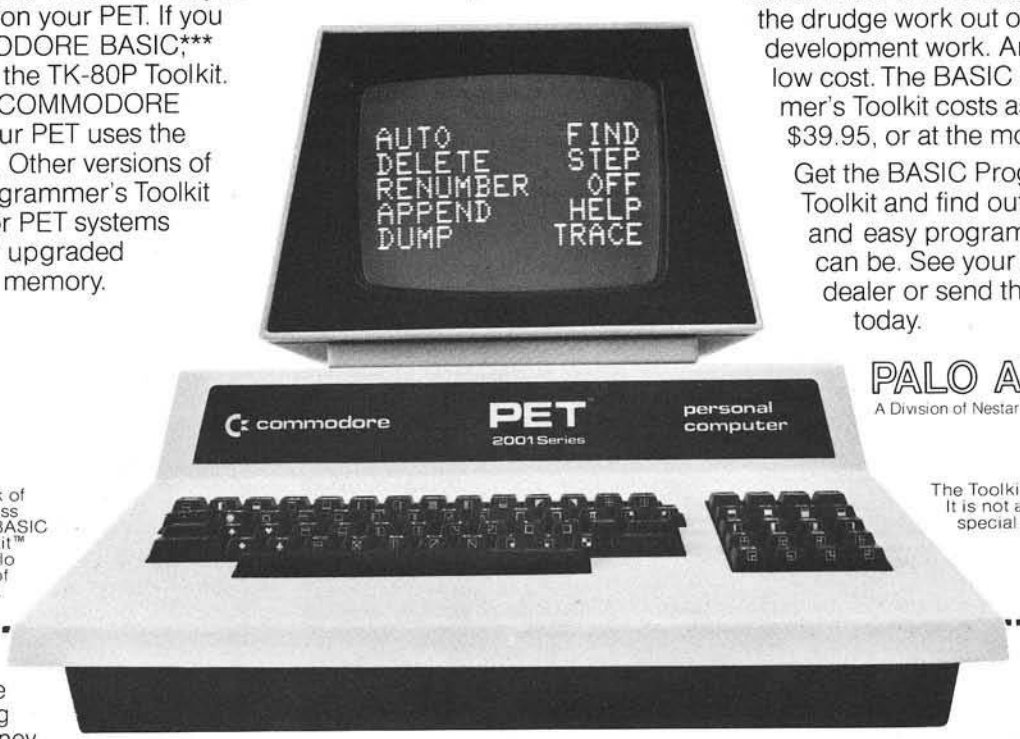
FIND locates and displays the BASIC program lines that contain a specified string. If you were to type *FIND A$,100–500,* your PET's screen would display all lines between line numbers 100 and 500 that contain *A$.*

RENUMBER renumbers the entire program currently in your PET. You can instantly change all line numbers and all references to those numbers. For instance, to start the line numbers with 500 instead of 100, just use *RENUMBER 500.*

HELP is used when your program stops due to an error. Type *HELP,* and the line on which the error occurs will be shown. The erroneous portion of the line will be indicated in reverse video on the screen.

These simple commands, and the other seven listed on the screen, take the drudge work out of program development work. And for a very low cost. The BASIC Programmer's Toolkit costs as little as $39.95, or at the most, $59.95.

Get the BASIC Programmer's Toolkit and find out how quick and easy program development can be. See your local PET dealer or send this coupon in today.



PALO ALTO ICs
A Division of Nestar Systems, Incorporated

PET™ is a trademark of Commodore Business Machines, Inc. The BASIC Programmer's Toolkit™ is a trademark of Palo Alto ICs, a division of Nestar Systems, Inc.

The Toolkit is fully assembled. It is not a kit and requires no special tools to install.

I want to save programming time and money.
Send me The BASIC Programmer's Toolkit that will give my PET 10 new and useful commands.
Fill in the appropriate line below:

Qty._____ TK-160 Toolkit(s) @ $39.95

Qty._____ TK-80P Toolkit(s) @ $59.95

Quantity discounts available to qualified dealers. Want to charge it? Call (415) 493-TOOL, or fill out the form below.

Enclosed is a ☐ money order ☐ check

(If charging): ☐ Bill VISA ☐ Bill Master Card.

Charge Card_____ Exp. Date_____

Master Card Interbank Number_____

Signature_____

Please include the amount of the Toolkit, plus $2.50 for shipping and handling. Please allow 4–6 weeks for delivery. SATISFACTION GUARANTEED, OR SEND IT BACK WITHIN 10 DAYS OF RECEIPT AND PALO ALTO ICs WILL REFUND YOUR MONEY.

SEND TO:

Name_____

Address_____

City_____ State_____ Zip_____

Phone_____

MAIL TO: Palo Alto ICs
A Division of Nestar Systems, Inc.
430 Sherman Avenue
Palo Alto, CA 94306
(415) 493-TOOL

For Multiple File Tape Backups

This PET utility takes the drudgery out of making multiple backup copies of cassette files.

G.R. Boynton
Dept. of Political Science
University of Iowa
Iowa City, IA 52242

Always make a backup copy! That is good advice, and I followed it assiduously for a year. Then it became apparent that something had to give. I had over 100 original programs or data files on separate tapes; which meant that I had over 100 backups. That made me one of the more regular customers of the store from which I buy tapes, and I had a very large sum of money tied up in tapes. In addition, my tapes were running me out of house and home. Something had to be done!!

Aside from lethargy, what kept me from doing something about this situation was the recognition of how slowly the PET tape drive operates, and the amount of time I projected it would take to make multiple file backup tapes or to use them once they were constructed. But I remembered a couple of programs that used fast forward to go skittering across the tape until the correct file was found so I dug them out and began to construct multiple file backup tapes. Everything was going fine until I had to save a program that took 2K of memory and one that took 11.3K of memory. Because of the way the program worked that wasted a lot of space for the first program and was not big enough for the second. The problem: a fixed amount of tape reserved for each program. That seemed a very unhandy way for a program to operate so I wrote this program called TAPE BACKUP.

TAPE BACKUP is designed to facilitate creating multiple file backup tapes. In order to do this effectively you need a long tape; a 46 minute or 60 minute tape will do. The first program saved on the tape is TAPE BACKUP. After TAPE BACKUP is loaded you are asked whether you want to use cassette #1or #2. Then a "table of contents" is displayed, and you are instructed to push F.FWD and then indicate the number of the file you want to access (or the location at which you wish to save a file). Once you indicate the file, the cassette fast forwards to the appropriate place and stops. The operation is very simple, and F.FWD is fast, at least relative to PLAY. In addition, the program gives instructions on what has to be changed in the program when you add a file to the tape to bring it up to date, and it will even compute the number of jiffies required to fast forward over a program of a given size.

Next I would like to describe the program and how it does what it does. Then I will suggest some simple procedures for using the program.

The program is designed to be quite flexible. It will run on machines with either the new or old PET BASIC ROMs, and it will permit using either cassette. The PEEK statement in line 90 determines whether it is the new or old ROM. The next few lines print an introduction to the program and determine which cassette will be used. Lines 180-195 define a variable "RO" which combine those two pieces of information, and this variable is used later in three GOSUB statements.

Lines 200 through 350 put the instructions to push F.FWD and the table of contents on the screen.

Lines 400 through 550 plus three subroutines and a data statement are the heart of the program. There are four tasks to be performed in this segment of the program.

1. Stop the cassette.
2. Ascertain the file to be accessed.
3. Determine the number of jiffies needed to get to that file.
4. Start the cassette; let it run the required length of time; and then stop the cassette.

Lines 400, 410, and 530 obtain the number of the file to be accessed. They constitute a loop which includes the GOSUB statement in 420. Depending on "RO", which specifies the ROM and the cassette being used, statement 420 sends the program to a statement which does two things. It sets the tape drive to an "off" status, and it stops the operation of the appropriate cassette motor. For the old ROM, memory location 519 sets the status of the first tape and 520 sets the status of the second cassette. For the new ROM the comparable memory locations are 249 and 250. If 52 is POKEd to 519 for the old ROM or 1 is POKEd to 249 for the new ROM this sets the status of tape 1 as off. For tape 2 the memory location 520 must be POKEd 1 for the old ROM and 250 must be POKEd 1 for the new ROM. Memory location 59411 is used for the first cassette motor in both old and new ROM's, and 59456 is used for the second

cassette motor. By POKing 61 for the first cassette or 223 for the second cassette, the motor is turned off.

The third task is accomplished by lines 440 through 500. In 440 the file number is changed from a string variable to a number. This allows the for/next loop in 450 through 490 to operate the appropriate number of times. Line 460 reads the Kth number on a data statement; the data statement is in line 888. Line 470 adds the past value of J (represented by J1) to 150 (which is the jiffies needed for the leader) to the number of jiffies needed for the file. If one wants the fourth file the loop will operate four times. The first time it will add the past value of J (which is zero) to 150 (for the leader) to the number of jiffies for the first file. And it continues in this way through four iterations. Thus, the loop calculates the number of jiffies needed to reach the file to be accessed.

The final task is fast forwarding to the appropriate place on the tape. This is done in lines 510 through 540 and the associated subroutines. The subroutine in 1100 (or whichever of the four is appropriate) sets a value of TS which is the current value of TI (the PET's clock) plus J. Then it starts the cassette motor. In 530 the value of TI is compared to TS; as long as TI is less than TS the cassette is fast forwarding. When TI is no longer less than TS the subroutine shuttled to by line 540 stops the cassette motor.

From line 600 on the program gives instructions. After the cassette stops one can either access the file or save a new file. The program contains two features for assisting in creating a new file. First, it will remind the user about the changes that should be made in TAPE BACKUP when a new file is added. Then it will compute the jiffies needed for a file of a given size. The user has to know how much memory is required for the program or the data set. This can be determined using FRE(0) when the program is loaded or before and after a data file has been read by a program.

Using the program is quite simple. It works particularly smoothly

```
10 REM *** TAPE BACKUP PROGRAM
20 REM *** G. R. BOYNTON
30 REM *** DEPARTMENT OF POLITICAL SCIENCE
40 REM *** UNIVERSITY OF IOWA
50 REM *** IOWA CITY, IOWA 52242
90 RM=PEEK(50003)
100 PRINT"[CLR][DN][DN][DN]";TAB(10);"*** TAPE BACKUP ***"
110 PRINT"[DN][DN][DN]THIS PROGRAM IS DESIGNED TO FACILITATE"
120 PRINT"CREATING MULTIPLE FILE TAPE BACKUPS."
130 PRINT"[DN]IT CAN BE USED EITHER TO SAVE A FILE"
140 PRINT"AT A PARTICULAR SPOT ON THE TAPE OR TO"
150 PRINT"ACCESS A PROGRAM OR DATA SET QUICKLY."
160 PRINT"[DN][DN][DN]DO YOU WANT TO USE TAPE #1 OR TAPE #2?"
170 PRINT"[DN][RC][RC][RC](RESPOND '1' OR '2')  ";:INPUT TD
180 IF RM=0 AND TD=1 THEN RO=1
185 IF RM=0 AND TD=2 THEN RO=2
190 IF RM=1 AND TD=1 THEN RO=3
195 IF RM=1 AND TD=2 THEN RO=4
200 PRINT"[CLR][DN][DN][DN]PRESS F.FWD KEY ON CASSETTE TAPE NOW."
210 PRINT"[DN]NEXT ENTER THE DESIGNATION FOR THE FILE"
220 PRINT"TO BE SELECTED."
230 PRINT"[DN][DN]THE FILES ON THIS TAPE ARE:"
240 PRINT"[DN] N   DATA FILES FROM CALENDAR"
250 PRINT"---   ----------"
260 PRINT"[DN]0.  FIRST FILE"
270 PRINT"1.  SECOND FILE"
280 PRINT"2.  THIRD FILE"
290 PRINT"3."
300 PRINT"4."
310 PRINT"5."
320 PRINT"6."
330 PRINT"7."
340 PRINT"8."
350 PRINT"9.  TENTH FILE"
400 N$=""
410 GET N$
420 ON RO GOSUB 1000,1010,1020,1030
430 IF N$="" THEN GOTO 410
440 N=ASC(LEFT$(N$,1))-48
445 IF N=0 THEN 600
450 FOR K=1 TO N
460 READ D(K)
470 LET J=J1+150+D(K)
480 LET J1=J
490 NEXT K
500 RESTORE
510 PRINT"SEARCHING FOR FILE ";N
520 ON RO GOSUB 1100,1110,1120,1130
530 IF TI<TS THEN 530
540 ON RO GOSUB 1200,1210,1220,1230
600 PRINT"[CLR][DN][DN][DN]FILE ";N;" HAS BEEN FOUND."
610 PRINT"[DN]PRESS 'STOP' ON THE CASSETTE."
620 PRINT"[DN]NOW YOU MAY ACCESS THE FILE NORMALLY"
630 PRINT"[DN]OR"
640 PRINT"[DN]SAVE A NEW FILE ON THE TAPE AT"
650 PRINT"[DN]THIS POINT."
660 PRINT"[DN][DN][DN]WOULD YOU LIKE INSTRUCTIONS FOR UPDATING"
670 PRINT"THIS PROGRAM WHEN ADDING A NEW FILE?"
680 PRINT"[DN]";:INPUT A$
690 IF A$="YES" THEN GOSUB 1300
700 PRINT"[CLR][DN][DN][DN]WOULD YOU LIKE TO COMPUTE THE NUMBER"
710 PRINT"OF JIFFIES REQUIRED FOR THIS FILE?"
720 PRINT"[DN]";:INPUT A$
730 IF A$="YES" THEN GOSUB 1400
740 PRINT"[DN]THAT'S IT."
750 END
888 DATA 325,700,5
1000 IF PEEK(519)=0 THEN POKE 519,52:POKE 59411,61:RETURN
1010 IF PEEK(520)=0 THEN POKE 520,1:POKE 59456,223:RETURN
1020 IF PEEK(249)=0 THEN POKE 249,52:POKE 59411,61:RETURN
1030 IF PEEK(250)=0 THEN POKE 250,1:POKE 59456,223:RETURN
1100 POKE 59411,53:TS=TI+J:RETURN
1110 POKE 59456,207:TS=TI+J:RETURN
1120 POKE 59411,53:TS=TI+J:RETURN
1130 POKE 59456,207:TS=TI+J:RETURN
1200 POKE 59411,61:RETURN
1210 POKE 59456,223:RETURN
1220 POKE 59411,61:RETURN
1230 POKE 59456,223:RETURN
1300 PRINT"[CLR][DN][DN][DN]TWO PARTS OF THE PROGRAM SHOULD BE"
1310 PRINT"UPDATED."
1320 PRINT"[DN]THE TABLE OF CONTENTS IN LINES 230"
1330 PRINT"THROUGH 350 SHOULD BE CHANGED BY ADDING"
1340 PRINT"THE NAME OF THE FILE ADDED."
```

```
1350 PRINT"[DN]THE DATA STATEMENT IN LINE 888 SHOULD"
1360 PRINT"HAVE THE NUMBER OF JIFFIES IN THE NEW"
1370 PRINT"FILE ADDED TO IT."
1380 PRINT"[HM]";:FOR K=1 TO 22:PRINT"[DN]";:NEXT K
1385 PRINT"PRESS SPACE BAR TO CONTINUE"
1390 GET A$:IF A$="" THEN 1390
1399 RETURN
1400 PRINT"[CLR][DN][DN][DN]THE NUMBER OF JIFFIES REQUIRED TO"
1410 PRINT"FAST FORWARD OVER A FILE DEPENDS ON THE"
1420 PRINT"SIZE OF THE FILE.  IF THE FILE REQUIRES"
1430 PRINT"2.1K OF MEMORY IT WILL TAKE A SMALL"
1440 PRINT"NUMBER OF JIFFIES. IF THE FILE REQUIRES"
1450 PRINT"11.3K OF MEMORY MORE JIFFIES WILL BE"
1460 PRINT"REQUIRED."
1470 PRINT"[DN][DN]HOW MANY BYTES OF MEMORY ARE REQUIRED"
1480 PRINT"FOR THE FILE?"
1490 PRINT"[RC](PLEASE GIVE A NUMBER LIKE 2.1 OR 11.8)"
1500 PRINT"[DN]";:INPUT M
1510 MM=M*1000
1520 JJ=MM/15
1530 PRINT"[DN]";JJ;" IS THE NUMBER OF JIFFIES"
1540 PRINT"[DN][DN]WOULD YOU LIKE TO COMPUTE THE NUMBER"
1550 PRINT"OF JIFFIES FOR ANOTHER FILE?"
1560 PRINT"[DN]";:INPUT A$
1570 IF A$="YES" THEN GOTO 1470
1580 RETURN
```

[CLR]	CLEAR SCREEN	[LC]	CURSOR LEFT
[DN]	CURSOR DOWN	[RC]	CURSOR RIGHT
[UP]	CURSOR UP	[RV]	REVERSE
[HM]	CURSOR HOME	[RVOFF]	REVERSE OFF

with two cassettes, one for tapes to be copied from and one for the tape to be copied to. However, I will not assume two cassettes are available in these instructions.

First, one needs a long tape with TAPE BACKUP saved as the first file. Load and run the program. DO NOT REWIND THE TAPE. The tape is now ready for saving your backup file '0'. Take the backup tape out of the cassette and put the tape to be copied from in the cassette and load that program. Once the program is loaded type PRINT FRE(0). That will give the amount of RAM left. If that number is subtracted from the RAM available you then know the amount of RAM used by the program. Take the program tape out and insert the backup tape which is still at the position it was after loading TAPE BACKUP. Save the program at this point. Rewind the tape and load TAPE BACKUP again. Add the name of the new file to the table of contents. Then type RUN 1400 to compute the jiffies needed, and add this to the data statement in line 888. To save a second program run TAPE BACKUP again asking for file '1'. The program will fast forward over your '0' file and be in place for saving file '1'. Then repeat the steps outlined above.

To access a file is even simpler. Load TAPE BACKUP. Specify the file you want to access. The program will fast forward to that file, and you load the program.

To save or access a data file (as opposed to a program) you need a program that will read and write the data file. Load TAPE BACKUP. When it has fast forwarded to the appropriate location take the backup tape out and load the program to read and write the data file. Read the data file, and take that tape out. Put the backup tape in the cassette, and write the data file.

Making multiple file backup tapes is always a rather boring task; it is one of the overhead costs of having a tape based system. But this program takes a good deal of the drudgery out of the task.

PET Vet

by Loren Wright
PET Specialist
MICRO Staff

Documentation on the PET is definitely improving, both from Commodore and from independent sources. PET veterans remember all the trouble they had in getting information, and I won't go into a long dissertation on Commodore's past inadequacies in this area. Neophytes can be assured that they won't have any of the same problems.

Commodore currently has available *PET User Manual (BASIC 2.0, $9.95), PET User Manual (BASIC 3.0, $9.95), CBM User Manual (BASIC 3.0, $9.95)* and *CBM User Guide ($7.95).* As this column goes to press, a manual covering the 4.0 BASIC *(Point of Reference Manual)* has just been released. When the magazine reaches you, a manual covering the 5¼" Floppy Disk should be available. Neither was available in time for this column, but both sound as if they would be valuable.

Recently several books applicable to the PET have appeared independently of Commodore. For the PET user who is new to both BASIC programming and the PET, *Hands-On BASIC with a PET* by Herbert D. Peckham (McGraw-Hill, 1979, $10.95) is an excellent starting point. Each chapter begins with specific learning objectives, progresses to a hands-on discovery session, then a discussion, and finally a quiz. Thinking is encouraged by "What do you think will happen if...?" Many illustrations of the PET's output and cartoons are liberally used to convey important points.

PET/CBM Personal Computer Guide by Carroll S. Donahue and Janice K. Enger (Osborne/McGraw-Hill, 1980, $14.95) is the comprehensive reference manual everyone hoped would come from Commodore. It *is* Commodore-authorized, which means, I imagine, that Commodore provided a lot of information and reviewed the manuscript. It is clearly written, well-organized, and prolifically illustrated. There are many programming hints, although the book is not intended as a programming manual.

The authors make liberal use in the text of photos, keyboard diagrams, flow charts, and actual PET output. All of the essential charts and tables (memory maps, ASCII and PEEK/POKE codes, error messages, and lots more) are included in an appendix. Whenever there is a difference between operation with old and upgrade ROMs this is clearly pointed out. The book lacks interfacing information, and assembly language is only touched upon, but otherwise it is an invaluable reference for every PET owner. The book is currently being revised and expanded to cover all three BASICs and all of the Commodore peripherals. The new edition, due in November, will be enclosed with all new PETs sold.

PET and IEEE 488 Bus (GPIB) by Eugene Fisher and C.W. Jensen (Osborne/McGraw-Hill, 1980, $14.95) covers just about everything you need to know about the GPIB. Even if you only want to understand the bus on an intuitive level, the book is designed to accomodate you. Of course, if you want timing diagrams and schematics, those are here too. Once you start reading it, you will probably find youself learning more than you had originally intended. The cartoons, aside from entertaining, actually reinforce the essential concepts. An especially valuable feature of the book is a section giving detailed instructions for using five different peripherals with the PET on the bus. Another is a section on adapting non-standard devices to the bus. Many PET owners fail to realize the potential of that mysterious connector back there between the power cord and the parallel user port. This book should get many reluctant users to actually use it!

Another book — *PET Machine Language Guide* — is available from Abacus Software (P.O. Box 7211, Grand Rapids, MI 49510, $6.95 plus $.70 postage). It covers a wide range of necessary operations, such as arithmetic, address handling, and input/output. Also included are detailed descriptions of many of PET's own routines.

Some Common BASIC Programs, PET edition by Poole, Borchers and Donahue (Osborne/McGraw-Hill, $12.50), should be valuable, too. Unlike other books of BASIC programs available, this book contains programs that will work in your PET without having to be converted from some other kind of BASIC. Also, the programs are available on 5¼" floppy disk and cassette.

With the increased attention to documentation on the part of Commodore, and with the arrival of these excellent independent works, the PET is now finally being well served, and it looks as if things are going to be getting even better.

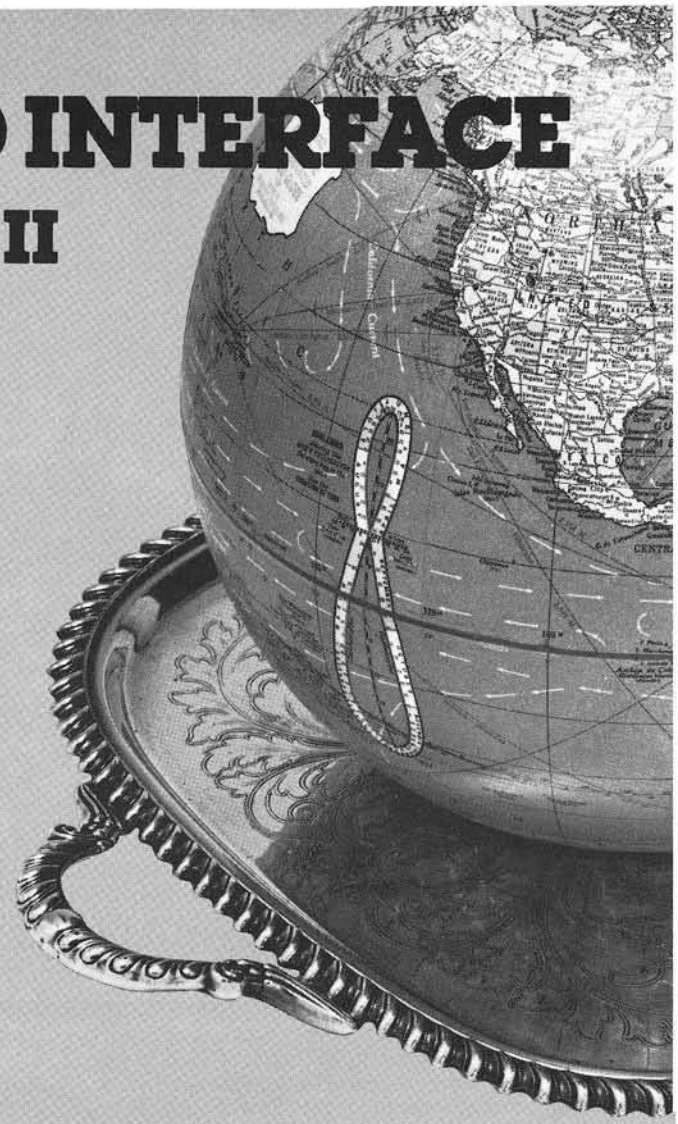
About Reviews

MICRO's standard policy is for independent reviewers to review products and books for the monthly MICROScope section. Manufacturers and publishers wishing to have their product(s) reviewed should send for and complete a Product Evaluation Form for each product submitted. Game-type software is not accepted for review. Readers interested in becoming potential reviewers should send for and complete a Reviewer Qualification form. Unsolicited reviews will not be accepted.

On a Related Subject

Robert Purser has announced that he will no longer review PET software in *Purser's Magazine*, citing as the primary reasons his diminishing PET readership and Commodore's announced priority shift from home to business systems.

REAL WORLD INTERFACE
FOR YOUR APPLE II



A/D + D/A

Commercial, scientific, and industrial data acquisition and control functions are now practical with Mountain Hardware's A/D + D/A card. Superfast conversion time permits high frequency and other applications not possible with slower cards.

A/D + D/A features:

- *Single PC card
- *16 channels analog to digital input
- *16 channels digital to analog output
- *9 μs conversion time
- *8 bit resolution
- *I/O cable assembly available
- *Operating manual contains sample applications with schematics, parts list, and guides for easy start-ups.
- *Self-test diagnostic software

If you've got a data acquisition or control application, Mountain Hardware has the answer with A/D + D/A. Drop by your Apple dealer and put your world on a silver platter.

Mountain Hardware
LEADERSHIP IN COMPUTER PERIPHERALS
A Division of Mountain Computer, Inc.
300 Harvey West Blvd., Santa Cruz, CA 95060
(408) 429-8600

Send me information.

Name _____

Address _____

City _____ State _____

Zip _____ Phone _____

*Apple II is a trademark of Apple Computer, Inc.

SPECIAL INTRODUCTION
ONLY $99.95

FOR THE FINEST PARALLEL INTERFACING ELEMENT FOR THE PET/CBM*

THE P.I.E.-C IS A VERY COMPLETE INTERFACE. IT COMES IN AN ATTRACTIVE CASE AND MOUNTS DIRECTLY TO THE COMPUTER, THUS ELIMINATING THE NEED FOR EXTRA SPACE IN THE BACK FOR BOXES AND MESSY PILES OF WIRE. YOUR SYSTEM CAN LOOK PROFESSIONAL. THE P.I.E.-C HAS COMPLETE ADDRESS SELECTABILITY OF DEVICE NUMBERS 4 THROUGH 30. AND, THERE'S NO NEED TO DISASSEMBLE YOUR UNIT TO CHANGE THE ADDRESS; IT'S AS SIMPLE AS SETTING THE DIP SWITCH. THIS SELECTABILITY ALLOWS BOTH YOUR LETTER-QUALITY PRINTER (NEC SPINWRITER, ETC.) AND YOUR GRAPHICS PRINTER TO BE CONNECTED AT THE SAME TIME BUT USED SEPARATELY BY ADDRESSING EACH WITH A DIFFERENT DEVICE#. NOW THAT'S REAL EFFICIENCY!!

THE P.I.E.-C WILL CONVERT NON-STANDARD PET CODES TO TRUE ASCII CODE. IT CAN BE SWITCHED IN OR OUT OF SERVICE AS NEEDED. FOR INSTANCE, IF YOUR SOFTWARE DOES THE CONVERSION, JUST SWITCH THE CONVERTER OFF. IF YOU NEED THE CONVERSION FOR PRINTING FROM BASIC, JUST SWITCH IT ON.

OUR INTERFACE PROVIDES EXTENSION OF THE IEEE-488 PORT ON THE PET/CBM. YOU DON'T NEED TO BUY OTHER CABLES BECAUSE OUR INTERFACE USES THE SAME CARD EDGE TYPE AS THE COMPUTER. THUS THE CABLE FROM THE FLOPPY DISC TO THE PET/CBM WILL NOW CONNECT INTO THE INTERFACE.

THE P.I.E.-C IS INTERCONNECTED WITH, AND POWERED BY, THE PRINTER USING A 3' DATA CABLE SUPPLIED WITH THE INTERFACE. THIS MEAN THAT THERE IS NO POWER SUPPLY NECESSARY AS WITH SERIAL INTERFACES. THE +5V IS SUPPLIED ON PIN# 18 OF THE CONNECTOR USED BY ALL TRUE CENTRONICS STANDARD INTERFACED PRINTERS. THIS INCLUDES THE CENTRONICS 779, P1, AND MANY OTHERS INCLUDING THE ANADEX PRINTERS.

OUR INTERFACE GOES BEYOND THE CENTRONIC'S COMPATIBILITY. THEREFORE, IT WILL OPERATE THE ANDERSON-JACOBSON AJ841 AND THE "PAPER TIGER" BY INTEGRAL DATA SYSTEMS. IN FACT, IT WILL DRIVE ANY PARALLEL-INPUT PRINTER THAT USES 8 DATA BITS AND 2 HANDSHAKING LINES.

IF YOU ARE INTERESTED IN THE MOST INTERFACE FOR YOUR MONEY, THEN PLACE YOUR ORDER TODAY. OUR INTRODUCTORY PRICE FOR THE P.I.E.-C INTERFACE WON'T LAST LONG. SO, DON'T DELAY . . . YOU MAY HAVE TO PAY THE LIST PRICE OF $119.95.

PLEASE SPECIFY YOUR PET/CBM TYPE (NEW OR OLD ROMS) AND THE BRAND AND MODEL OF THE PRINTER YOU WILL BE USING. WE ATTACH THE CORRECT CONNECTOR ON THE CABLE. ALL ORDERS MUST BE PREPAID OR COD CASH. ADD $5 FOR SHIPPING AND HANDLING. MARYLAND RESIDENTS ADD 5% SALES TAX.

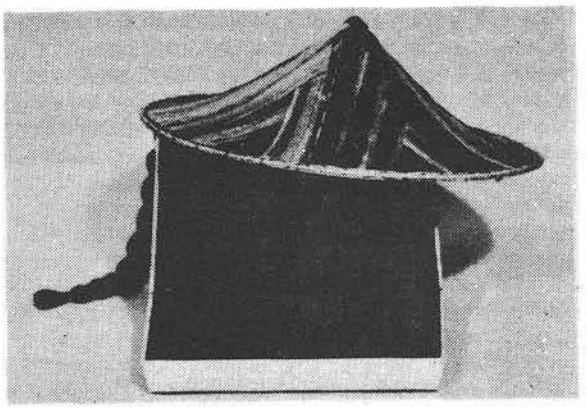
LEMDATA PRODUCTS, P.O. BOX 1080, COLUMBIA, MD. 21044

PHONE (301) 730-3257

*PET/CBM are trademarks of Commodore Business Machines

WP-6502

a very fine word processor



for OHIO SCIENTIFIC

Tape (C1,C2,C4) $75	8" 65D & 65U $125
5" Disk (C1,C2,C4) ...$75	Descriptive
8" Disk for 65D $75	Brochure ..... **FREE**

Dwo Quong Fok Lok Sow
23 East 20th Street
New York City, New York 10003
(212)685-2188

THE WIZARD AND THE PRINCESS
HI-RES ADVENTURE #2

Only ON-LINE SYSTEMS could deliver a HI-RES ADVENTURE game on such an epic scale. In this adventure you find you must do battle against an evil wizard in order to save the life of the princess. To find the wizard and his castle you must first cross deserts, oceans, mountains, travel to an island and encounter many strange beasts. You will be forced to learn magic, navigate at sea and dig for treasure. This game should provide months of adventure.

- HUNDREDS OF HI-RES PICTURES (looks great on b/w and color televisions)
- FULL 21-COLOR!! HI-RES GRAPHICS (each room a work of art)
- YOUR GAME MAY BE SAVED FOR LATER CONTINUANCE
- RUNS ON BOTH 48K APPLE II AND APPLE II PLUS
- BY FAR THE MOST AMBITIOUS GRAPHIC GAME EVER WRITTEN FOR THE APPLE!!

Hi-Res Adventure #2 is available now at your local computer store and requires a disk drive. To order directly send $32.95 to:

On-Line Systems
36575 Mudge Ranch Road
Coarsegold, CA 93614
209-683-6858

VISA, MST CHG, COD, CHECK ACCEPTED

Look for Hi-Res Football coming soon

OHIO SCIENTIFIC'S

In the October issue of the Ohio Scientific Small Systems Journal, we are introducing the C1P family of personal computers — the C1P Series 2. The article describes the product and additionally contains some applications information.

Another new product being featured this month is the DMS Planner. The Planner is an extremely sophisticated data management and planning system. The article contains several examples which illustrate some of the system's remarkable features.

As always, comment on article content is welcomed. Please submit suggestions, or any other contributions to:

Ohio Scientific, Inc.
1333 S. Chillicothe Road
Aurora, Ohio 44202

C1P Series 2 Computers

Ohio Scientific's new C1P Series 2 line is the most cost effective of the Ohio Scientific family of microcomputers. In spite of its economical price, the new C1P includes many deluxe features usually found only in much more expensive systems.

The features of the standard C1P Series 2 include:
- 8K BASIC-in-ROM and 8K program RAM
- Switch selectable audio cassette/modem/printer port
- Sound, music and voice output via digital to analog converter
- Software selectable display format (24x24 or 12x48)

The basic system is easily expandable to single or dual mini-floppies and 32K of program RAM. Additional expansion via the new 630 board allows expansion to a color display, joystick interface, AC remote control and home security system.

The C1P Series 2 personal computer is specifically designed for the first-time personal computer user and for use in educational environments.

Personal or Home Computers

Challenger 1P's advanced character graphics, noise-free display, programmable keyboard and high speed BASIC make it capable of spectacular video games, animation, and elaborate computer games. Ohio Scientific offers an extensive library of one and two player video games that are very similar to conventional "arcade games," as well as a complement of "standard" computer-type games.

The C1P Series 2 floating point decimal arithmetic capability in conjunction with its cassette storage abilities make it practical for many forms of personal financial aid and analysis. Ohio Scientific's cassette library includes a checkbook balancing program, savings account program, and annuity and loan analysis programs. Budget planning aids include home ownership cost analysis and expense accounting. A complete home budget system is available for use on the C1P Series 2 mini-floppy system.

It should be pointed out that a mini-floppy disk is a necessity for some of the advanced applications mentioned above.

Additionally, the C1P Series 2 BASIC has full transcendental arithmetic capability, including trigonometric functions, logarithms, exponentiation, and full scientific notation. These features are available in the "immediate mode" of operation as well as the stored program mode. For instance, a user can quickly turn the computer on, type in an equation as a single line, and press return to get an answer. The computer can double as an advanced scientific calculator with much greater ease of use than any available calculator.

The program storage and alphanumeric capability of the Challenger 1P make it extremely valuable to engineers, students, and educators for solving scientific,engineering and mathematical analysis problems. Ohio Scientific's cassette library includes several advanced mathematics oriented programs including a programmable calculator simulator and a mathematical function library. The library also includes applications programs such as definite integrals, statistical analysis, and other complex mathematical functions. In general the Challenger 1P will be hundreds of times faster than the most powerful scientiific calculators in the "number-crunching" applications.

Education

Challenger 1P series personal computers are extremely versatile in educational computing applications. Once the user gets involved in the educational applications of these machines, he will quickly consider them a necessity in the educational process.

Young children from kindergarten to grade six are expecially attracted to computers. As the child's reading ability develops he quickly masters the elementary operations of the computer. It is not at all unusual for six-year old children to respond to mathematical problems on a personal computer. Children's natural fascination with computers in conjunction with C1P's cartoon-like interactive capability make the computer highly valuable in a modern educational environment. Programs which teach, tutor and drill students in virtually all areas of education can be easily program med on the Challenger 1P system. Ohio Scientific has a full library of several types of educational games which can be used as an example in programming such applications. These programs range from a simple "Sesame Street" type arithmetic cartoon through mathematical drills, to word games such as "Hangman".

Another broad area of education is in teaching computer fundamentals. The Challenger 1P utilizes the most popular upper level language, BASIC, in a very complete and concise implementation. With the Challenger 1P the user can teach or learn BASIC in conjunction with any of the commonly available text books on the BASIC programming language. The C1P series machines have full machine code accessibility including the machine code monitor so that advanced students can enter, edit and execute machine programs. A very fast and interactive assembler/editor is available to run on Challenger 1P machines so that students can be introduced to the concepts of assembler programming and editing.

Advanced Applications

There are many other applications of the basic 1P machines that have not been mentioned here. The C1P Series 2 mini-floppy provides the user with the extra convenience of virtually instantaneous loading and storing of programs on mini-floppy disks. The addition of a mini-floppy disk drive to the C1P also provides convenient construction and access of data files. Using the file capabilities of the C1P MF, an educator can develop an interactive textbook with a quick access data base for any educational topic. In the home, the data file operation of the mini-floppy makes the Challenger 1P a deluxe personal service computer giving the user easy access to phone numbers, a personal calendar, addresses and other file-type information.

C1P Series 2 Expansion

The 630 I/O Expander board is available for addition to either the C1P or C1P MF. This board provides the C1P with the state-of-the-art in input/output capabilities rivaling the most expensive small computer systems available today. This board allows direct interface with joysticks, remote keypads, AC remote control units, home security system and more. It also substantially enhances the video display capabilities of the Series 2 models in the Challenger 1P line by allowing the display of up to 16 colors with any of the standard 256 graphics characters. The color is available either as standard composite video or with RGB (red-green-blue) direct color drive.

MDMS PLANNER

Ohio Scientific, Inc. is pleased to introduce MDMS Planner, a preprogrammed computer system for the home computerist as well as managers and consultants.

Planner is a technical analysis computer system which was designed to perform financial planning statements and analysis projections. And because Planner is preprogrammed and very flexible, it is easy for the untrained computer user and non-technically oriented person to use. This flexibility is accomplished through the use of models.

A model is a small scale representation of a large scale event. Models are used throughout industry in far too many forms to mention, but a common and familiar type of model is the wind tunnel.

The wind tunnel has made possible great achievements in the field of aerodynamics, because of three characteristics it shares with other types of models: precision, efficiency, and relatively low cost. Just think of the difference in working with a six-foot model as opposed to a full-sized aircraft. The model allows for more precise measurements of any variance; it can be adjusted more efficiently and simply than full scale systems, and it costs much, much less, both in intitial investment and redesign.

Financial modeling shares these desirable features with other types of modeling and is similar in concept. Alike the wind tunnel, financial modeling can be useful to everyone, from home budgeters to plant managers and consultants. Anyone who needs to know where money is going (or should be going!) can benefit from financial modeling.

Moreover, Planner provides a means of storing models and their results on disk, each stored in separate data files. This allows the user to recall a model and/or a given set of results for future use. Such a feature saves time by not having to re-enter models and/or results (or vice versa) since both are stored separately.

When constructing a model, the user must lay out and design it by defining the lines and columns that should exist. This includes specifying title lines (/T), blank lines (/), single (/-), and double underlining (/=) (Example 1 and 2). Example 1 illustrates lines that might be used in a home budget. After each line is entered, it is formatted and displayed to the right of its entry. Example 2 illustrates columns that might be used for the same home budget. Like Example 1, after each column is entered, it is formatted and reprinted.

Example 1

```
LINE EDIT: ^UP ^DOWN ^COPY ^MONTHS ^WIDTH ^EDIT ^<  ^>
-------------------------------------------------------------
BUDGET 1980/T                          BUDGET 1980
INCOME/T                                 INCOME
/-                                 --------------------------
SALARY                             SALARY                (4)
BONUS                              BONUS                 (5)
/-                                 --------------------------
TOTAL INCOME                       TOTAL INCOME          (7)
/
EXPENDITURES/T                          EXPENDITURES
/-                                 --------------------------
GROCERY                            GROCERY               (11)
UTILITIES                          UTILITIES             (12)
CAR PYMT & TRANS                   CAR PYMT & TRANS      (13)
HOUSE PYMT & MAINT                 HOUSE PYMT & MAINT    (14)
INSURANCE                          INSURANCE             (15)
SAVINGS                            SAVINGS               (16)
CLOTHING                           CLOTHING              (17)
MEDICAL                            MEDICAL               (18)
RECREATION                         RECREATION            (19)
 ----------------------------
 CURSOR AT LINE 1          ------------------------------------
                           FORMATS:  /T  /-  /=  /$  /%  /I
```

Example 2

```
COLUMN EDIT: ^LEFT ^RIGHT ^COPY ^MONTHS ^WIDTH ^EXIT ^<  ^>
-------------------------------------------------------------


JAN/80   FEB/80   MAR/80   APR/80   MAY/80   JUN/80   JUL/80

 AUG/80   SEPT/80


-------------------------------------------------------------
CURSOR AT C1                      FORMATS:  /L  /R  /$  /%  /I
```

Example 3

```
RULE EDIT: ^UP ^DOWN ^COPY ^EXIT ^< ^>
-------------------------------------------------
ADD L4 L5, SAVE L7
MUL L4 BY 12%, SAVE L11
MUL L4 BY  5%, SAVE L16
MUL L7 by 10%, SAVE L19
MUL L7 BY 10%, SAVE L20
MUL L7 BY 35%, SAVE L19, C6
MUL L7 BY 15%, SAVE L20, C6
MUL L4 BY 20%, SAVE L11, C12
MUL L7 BY 25%, SAVE L20, C12
TOT L11 THR L21, SAVE L23
DIV L23 BY L7, SAVE L24
SUB L23 FROM L7, SAVE L25
TOT C1 THR C12, SAVE C13



---------
RULE #1

-------------------------------------------------
OPERATIONS:  ADD,SUB,MUL,DIV,AVE,FIL,TOT,MAX,MIN,GRO
```

The initial values and amounts of the lines and columns are then entered. Finally, the calculations, or what is referred to as the rules to be performed on the lines and columns, are entered. Example 3 is a list of rules that might be run on the home budget example. For instance, rule two (MUL L4 BY 12%, SAVE L11" in English means "multiply line 4 by 12% and save the answer in line 11". A simpler interpretation is that the grocery bill will be 12% of each month's salary.

Planner is an unstructured system which allows the user to set up models and perform specific operations with a single key stroke. Models may be executed and set up in any fashion and can be edited and re-entered, allowing the user to run a "What If" pro forma calculation (Example 4), budgets (Example 5), etc. Because Planner is a semi-screen formatted system, formatted and easy to read reports and worksheets can be displayed on both the console and line printer.

Example 4: "What If" my sales for the first quarter were one million dollars, cost of sales were $530,000 and sales grew by 25% for the remaining three quarters, but the cost of sales only increased by 20%. What would gross profit be per quarter? Per year? (See copy of pro forma below.)

Example 4

```
This is an example of a "What If" pro forma calculation!
                                                                    12/31/79
                                                                    PAGE 1

                          1ST QTR     2ND QTR     3RD QTR     4TH QTR      ANNUAL
        INCOME
------------------------ ---------- ---------- ---------- ---------- ----------
SALES              (3)  1000000.00       0.00       0.00       0.00       0.00
COST OF SALES      (4)   530000.00       0.00       0.00       0.00       0.00
======================== ========== ========== ========== ========== ==========
GROSS PROFIT       (6)        0.00       0.00       0.00       0.00       0.00



This is an example of a "What If" pro forma calculation!
                                                                    12/31/79
                                                                    PAGE 1

                          1ST QTR     2ND QTR     3RD QTR     4TH QTR      ANNUAL
        INCOME
------------------------ ---------- ---------- ---------- ---------- ----------
SALES              (3)  1000000.00 1250000.00 1562500.00 1953125.00 2441406.25
COST OF SALES      (4)   530000.00  636000.00  763200.00  915840.00 1099008.00
======================== ========== ========== ========== ========== ==========
GROSS PROFIT       (6)   470000.00  614000.00  799300.00 1037285.00 1342398.25
```

Example 5: Now that I have received a raise, can the cost of leasing a car and transportation expenses (leasing and transportation cost estimated at $68.00/mo) fit in my budget? Keep in mind that groceries are 12% of my salary, utilities fluctuate according to seasons, savings are 5% of salary, recreation is 10% of income, and contributions and gifts are 10% of income. All other expenses are fixed amounts.

Note: Twice a year a $500 bonus is given: Vacation (June) and Christmas, thus causing more money to be needed for certain expenditures in June and December. (See copy of budget.)

Example 5

Bob & Nancy Jones
Monthly Budget
For 1980
(Which includes the additional expense of leasing a car)

12/31/79
PAGE 1

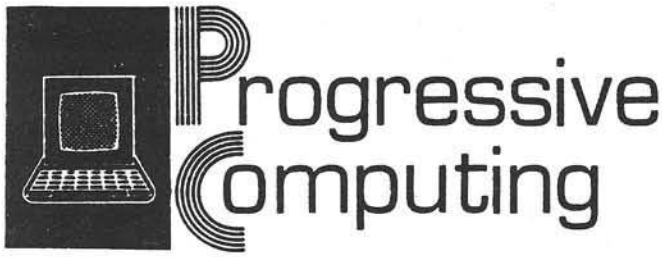
BUDGET 1980 INCOME		JAN/80	FEB/80	MAR/80	APR/80	MAY/80	JUN/80	JUL/80
SALARY	(4)	921.49	939.21	950.00	954.87	950.00	950.00	930.71
BONUS	(5)	0.00	0.00	0.00	0.00	0.00	500.00	0.00
TOTAL INCOME	(7)	921.49	939.21	950.00	954.87	950.00	1450.00	930.71
EXPENDITURES								
GROCERY	(11)	110.58	112.71	114.00	114.58	114.00	114.00	111.69
UTILITIES	(12)	170.00	170.00	155.00	140.00	125.00	110.00	95.00
CAR PYMT & TRANS	(13)	68.00	68.00	68.00	68.00	68.00	68.00	68.00
HOUSE PYMT & MAINT	(14)	215.00	215.00	215.00	215.00	215.00	215.00	215.00
INSURANCE	(15)	62.50	62.50	62.50	62.50	62.50	62.50	62.50
SAVINGS	(16)	46.07	46.96	47.50	47.74	47.50	47.50	46.54
CLOTHING	(17)	20.00	20.00	20.00	20.00	20.00	20.00	20.00
MEDICAL	(18)	25.00	25.00	25.00	25.00	25.00	25.00	25.00
RECREATION	(19)	92.15	93.92	95.00	95.49	95.00	507.50	93.07
CONTRIB & GIFTS	(20)	92.15	93.92	95.00	95.49	95.00	217.50	93.07
MISC.	(21)	20.00	20.00	20.00	20.00	20.00	50.00	20.00
TOTAL EXPENDITURES	(23)	921.45	928.01	917.00	903.80	887.00	1437.00	849.86
PERCENTAGE/INCOME	(24)	100.0%	98.8%	96.5%	94.7%	93.4%	99.1%	91.3%
UNBUDGETED MONEY	(25)	0.04	11.20	33.00	51.07	63.00	13.00	80.85

12/31/79
PAGE 2

BUDGET 1980 INCOME		AUG/80	SEP/80	OCT/80	NOV/80	DEC/80	ANNUAL
SALARY	(4)	953.00	967.18	942.26	983.64	997.77	11440.13
BONUS	(5)	0.00	0.00	0.00	0.00	500.00	1000.00
TOTAL INCOME	(7)	953.00	967.18	942.26	983.64	1497.77	12440.13
EXPENDITURES							
GROCERY	(11)	114.36	116.06	113.07	118.04	199.55	1452.64
UTILITIES	(12)	95.00	110.00	125.00	140.00	155.00	1590.00
CAR PYMT & TRANS	(13)	68.00	68.00	68.00	68.00	68.00	816.00
HOUSE PYMT & MAINT	(14)	215.00	215.00	215.00	215.00	215.00	2580.00
INSURANCE	(15)	62.50	62.50	62.50	62.50	62.50	750.00
SAVINGS	(16)	47.65	48.36	47.11	49.18	49.89	572.01
CLOTHING	(17)	20.00	20.00	20.00	20.00	125.00	345.00
MEDICAL	(18)	25.00	25.00	25.00	25.00	25.00	300.00
RECREATION	(19)	95.30	96.72	94.23	98.36	149.78	1606.51
CONTRIB & GIFTS	(20)	95.30	96.72	94.23	98.36	374.44	1541.18
MISC.	(21)	20.00	20.00	20.00	20.00	70.00	320.00
TOTAL EXPENDITURES	(23)	858.11	878.36	884.14	914.45	1494.16	11873.34
PERCENTAGE/INCOME	(24)	90.0%	90.8%	93.8%	93.0%	99.8%	1141.2%
UNBUDGETED MONEY	(25)	94.89	88.82	58.12	69.19	3.61	566.79

After reviewing the results of the model, the user can change any or all of these values. Immediately re-execute the model and instantly obtain new results. This process can be repeated as many times as needed. This feature is what makes Planner so astounding! It allows the user to actually see what future benefits or losses which could occur under the given set of circumstances.

Many business administrators spend much time forecasting the likely effects of business transactions for a distinct period of time. Planner provides the essentials to perform such forecasting along with profit and loss projections, budgets, make/buy decision analyses, cost extimating, etc. Planner uses are limited solely to the imagination of the user.



Hardware: C1P Video Mod: Makes your 600 Video every bit as good as the 4P and 8P. Gives 32/64 CHR/Line with guard-bands 1 and 2 Mhz. CPU clock with 300, 600 and 1200 baud for Serial Port. Complete Plans $19.95
KIT(Hardware and Software) $39.95
Installed:32CHR — $79.95, 64CHR — $84.95
Extra K of Video RAM for 64CHR. not included!

C1P Sound Effects Board: Completely programmable! For the discriminating hobbiest, the best board on the market for creating sound and music. Can be interrupt driven so that you can use it for gaming purposes. Has on board audio amp, 16 bit interval timer, 128 Bytes of RAM and two 8 bit parallel I/O ports. Assembled and tested $89.95
Bare Board $39.95 — both include Prog. Manual and Sample Software.

C1P Hi Speed Cassette Kit: Gives a reliable 300, 600, and 1200 Baud. No symmetry adjustments — the ideal fix for OSI's cassette interface. Easily implemented in 30 minutes. Will save you time and money even the first night you use it! $12.95

****SPECIAL INTRODUCTORY OFFER****

Programmable Character Generator Board $89.95
You can use OSI's characters or you can make your own. Imagine you can now do true high resolution graphics 512 x 256 individual dots in the 64x32 screen format. And all under your control!
Other mods available — send for catalog.

SOFTWARE (with Documentation)

PC CHess V1.9 $19.95
Play Chess against your computer!

Helicopter Pilot:(64 CHR Video Only) $8.95
An Excellent Graphics Program!

Golf Challenger $14.95
From 1 to 4 players. Play a round of golf on your 18 hole golf course. One of the best programs I have seen! You can even design your own course. Comes with full documentation (14 pages).

Two Very Intricate Simulations!

Wild Weasel II: You operate a Sam Missile base during a Nuclear War. Not as easy as you think! You must operate in a three dimensional environment.

Failsafe II: The shoe is on the other foot! Here you are in the attacking bomber and you must penetrate deep into enemy territory. Can you survive? An extremely complex electronic warfare simulation! SPECIAL, both for $19.95

Many, Many more. Send for Catalog with free program (Hard Copy) and BASIC Memory Map. $1.00. Two Locations to serve you:

3336 Avondale Court, Windsor, Ontario
Canada N9E 1X6
(519) 969-2500
or
3281 Countryside Circle, Pontiac TWP. MI 49057
(313) 373-0468

American Data, Inc.

The world's largest distributor of Ohio Scientific, Inc. Microcomputer Systems

ANNOUNCES

That OSI has appointed American Data as the exclusive distributor of Ohio Scientific products for Europe including the United Kingdom, Benelux, France, West Germany, Austria, Switzerland, Denmark, Italy, Spain, Portugal, Greece, Turkey, and Iceland.

Dealer inquiries invited. Contact Barbara Hall, 352-23-172, Luxembourg or David O'Brien, American Data (301) 840-9540 Telex 64405 (USA).

Paged Printer Output for the APPLE

Improve the format of your printed output by adding a page mode to your system.

Gary Little
#101-2044 West Third Ave.
Vancouver, B.C. V6J 1L5

If you have ever sent output to a printer you have probably become very annoyed when the output continued from the the very end of one page and then on to the next. Wouldn't it be nice if the printer would automatically scroll to the top of a new page when it got near the bottom of the previous one? Of course it would, and the PAGER program will do it for you. PAGER will count the number of line feeds that are sent by APPLE to the printer and when this total reaches 54, twelve blank lines will be generated to automatically bring you to the top of the next eleven-inch page. PAGER can be used from within a program or from immediate-execution mode. It is extremely useful for LISTing long programs page by page.

PAGER, the source listing of which appears below, has been written for use with a serial printer that is connected to the APPLE serial interface card. If PAGER is to be used in conjunction with a parallel printer connected to the APPLE parallel interface card, two bytes of the routine must first be changed. To make these changes, load PAGER, and then enter the following two commands from BASIC:

```
POKE 785,2
POKE 812,2
```

The modified program should then be saved.

In order to change the number of lines that are printed before PAGER causes the paper to scroll to the top of the next page, enter the command POKE 798,LP from BASIC, where LP is the required number of lines per page.

In order to change page length, enter POKE 804,PL from BASIC, where PL is six times the length of the page (in inches). For example, for an eleven inch page, PL = 66. Note that PL must be greater than LP.

Output to the printer can be stopped after each page is printed by entering a POKE 822,1 command before activating PAGER. To proceed after a page has been printed, simply press any key on the keyboard. This 'page pause' feature must be used when the user is feeding each piece of paper to the printer manually. To turn off the 'page pause', enter a POKE 822,0 command.

Instructions for Use Within a Program

The following sequence should be used to turn the printer on and off from within a BASIC program:

```
5   D$ = CHR$(4)
10   PRINT D$;"PR#1"
20   LW = 132 : REM LINE WIDTH
30   PRINT CHR$(9);LW;"N" : PRINT
CHR$(9); "K"
40   CALL 768 : REM TURN ON
PAGER
.
.   (Generate Output)
.
   50   PRINT D$;"PR#0" 0 : REM
TURN PRINTER OFF
```

If DOS is not being used, change line 10 to PR#1 and line 50 to PR#0 and delete line 5. If a serial printer is being used, delete lines 10, 20, and 30.

Instructions for Use Outside a Program

If a serial printer is involved, PAGER can be activated by a CALL 768 from BASIC. It can be deactivated by a PR#0. If a parallel printer is involved, PAGER can be activated by performing the following four steps:

1. Enter PR#1
2. Enter CTRL-I 132N (132 or other line width).
3. Enter CTRL-I K
4. Enter CALL 768

It can be deactivated by a PR#0.

Additional Notes:

1. Remember to set the DIP switches on the serial printer interface card for the appropriate baud rate and line width before activating PAGER.

2. Remember to adjust the paper in the printer so that the first line printed will be at the desired starting position before activating PAGER.

3. Make sure that a PRINTed line will not exceed the line width which has been set for the printer. If it does, then the overflow will appear on the next line and this line will not be taken into account by PAGER.

```
                        2     ********************************
                        3     *
                        4     * PAGER
                        5     *
                        6     * BY GARY LITTLE
                        7     * #101-2044 WEST THIRD AVENUE
                        8     * VANCOUVER B.C.  V6J 1L5
                        9     *
                        10    * INSTRUCTIONS:
                        11    *
                        12    * POSITION THE PAPER IN YOUR
                        13    * PRINTER AND THEN CALL 768
                        14    * FROM BASIC TO ACTIVATE
                        15    * THIS ROUTINE. TO DEACTIVATE,
                        16    * ENTER A PR#0.
                        17    *
                        18    * PAGE PAUSE FEATURE:
                        19    *    POKE 822,0 TO TURN OFF
                        20    *    POKE 822,1 TO TURN ON
                        21    *
                        22    * LINES PRINTED PER PAGE:
                        23    *    POKE 798,LP
                        24    *
                        25    * PAGE LENGTH:
                        26    *    POKE 804,PL
                        27    *
                        28    * DESCRIPTION:
                        29    *
                        30    * THIS ROUTINE WILL SEND 'PL-LP'
                        31    * BLANK LINES TO THE PRINTER
                        32    * AFTER 'LP' LINES HAVE BEEN
                        33    * SENT BY THE USER.
                        34    *
                        35    * DEFAULTS:
                        36    *    LP = 54
                        37    *    PL = 66 (11" PAPER)
                        38    *    PAGE PAUSE OFF
                        39    *
                        40    ********************************
                        41    COUNT     EQU  $6         LINE COUNT STORAGE
                        42    CSWL      EQU  $36        OUTPUT HOOK
                        43    DOS       EQU  $3EA       DOS I/O UPDATE HOOK
                        44    KBD       EQU  $C000      KEYBOARD
                        45    STRB      EQU  $C010      KEYBOARD STROBE
                        46    PRINT     EQU  $C100      PR#1 SERIAL OUTPUT
                        47              ORG  $300
0300: A9 0F             48              LDA  #<START    SET OUTPUT HOOK
0302: 85 36             49              STA  CSWL        TO START OF ROUTINE.
0304: A9 03             50              LDA  #>START
0306: 85 37             51              STA  CSWL+1
0308: A9 00             52              LDA  #$00       ZERO THE LINE COUNTER.
030A: 85 06             53              STA  COUNT
030C: 4C EA 03          54              JMP  DOS        GIVE NEW HOOK TO DOS.
030F: 48                55    START     PHA             ;ROUTINE STARTS HERE.
0310: 20 00 C1          56              JSR  PRINT      SEND CHARACTER TO PRINTER.
0313: 68                57              PLA
0314: C9 8D             59              CMP  #$8D       CARRIAGE RETURN?
0316: F0 01             60              BEQ  LINE       BRANCH IF IT IS.
0318: 60                61    NEXT      RTS
0319: E6 06             62    LINE      INC  COUNT      INCREMENT LINE COUNT.
031B: A5 06             63              LDA  COUNT
031D: C9 36             64              CMP  #$36       LINE COUNT = 54?
031F: D0 F7             65              BNE  NEXT       IF NOT, THEN RETURN.
0321: A5 06             66    BLANK     LDA  COUNT
0323: C9 42             67              CMP  #$42       PAGE LENGTH MET?
0325: F0 0A             68              BEQ  LOOP
0327: E6 06             69              INC  COUNT      INCREMENT THE COUNTER
0329: A9 8A             70              LDA  #$8A       LOAD A LINE FEED
032B: 20 00 C1          71              JSR  PRINT       AND SEND IT TO THE PRINTER
032E: 38                72              SEC
032F: B0 F0             73              BCS  BLANK
0331: A9 00             74    LOOP      LDA  #$00       ZERO THE COUNTER.
0333: 85 06             75              STA  COUNT
0335: A9 00             76              LDA  #$00       CHANGE TO LDA #$01 TO
0337: F0 08             77              BEQ  DONE        GET 'PAGE PAUSE'.
0339: 2C 00 C0          78    AGAIN     BIT  KBD        WAIT FOR KEYPRESS
033C: 10 FB             79              BPL  AGAIN       BEFORE CONTINUING.
033E: 2C 10 C0          80              BIT  STRB       CLEAR KEYBOARD STROBE.
0341: 60                81    DONE      RTS

--- END ASSEMBLY ---

TOTAL ERRORS: 0

66 BYTES GENERATED THIS ASSEMBLY
```

μ

MICRO SCOPE

Number 3

"Readings in Literature"

1. **Microcomputers which can use product:** Apple II or Apple II+.
2. **System hardware requirements:** Disk II and at least 32K of memory.
3. **System software requirements:** No special requirements.
4. **Product features:** The program is designed to assist in the study and memorizing of 27 well-known pieces of literature including prose, poems and songs. Such diverse items as "The 23rd Psalm," "The Bill of Rights," "Casey at the Bat," and several Shakespearean items are included. Graphics are used to excellent advantage to keep the various readings interesting and a "Hangman" type of "guess the word" system is used to assist in memorizing when necessary.
5. **Product performance:** Performance is very good and the product represents a novel approach to learning.
6. **Product quality:** The program seems very well written and no problems were encountered during extensive testing.
7. **Product limitations:** There are no unusual product limitations.
8. **Product documentation:** Very little written documentation is furnished; however, none is really required as the introductory instructions on the disk are very thorough.
9. **Special user requirements:** There are no special user requirements.
10. **Price/Feature/Quality evaluation:** The price of $24.95 seems reasonable in relation to the amount of programming involved and the overall high quality of the product.
11. **Additional comments:** It should be noted that the disk is "locked" to prevent unauthorized copies and thus backup copies cannot be made. However, the manufacturer does offer to replace any disk which becomes damaged for a $10.00 charge.
12. **Reviewer:** John B. Shanes Jr., P.O. Box 68, Mechanicsville, VA 23111. Manufactured by George Earl.

Biorhythm: An AIM BASIC Programming Excercise

Whether or not you believe in biorhythm theory, the programs presented here make an interesting demonstration and provide some useful techniques for using the AIM and its printer.

P.E. Burcher
7012 Stone Mill Place
Alexandria, VA 22306

The natural, regularly repeating rhythms of our world continually affect our daily lives. The sun and moon appear and disappear, the tides rise and fall and the seasons change. In all life, regular cycles are seen. Early in this century various researchers found evidence supporting the cyclic nature of human behavior. Through the years a body of "biorhythm" theory has evolved.

Physical, emotional and intellectual biorythm cycles begin at birth and continue throughout life. The rhythm of physical capacity covers 23 days, high during the first half of the cycle and low during the second. The first 14 days of the 28 day emotional or sensitivity cycle reflect cheerfulness and optimism followed by and emotionally negative period. Similarly, the cycle of intellectual capacity covers 33 days. In the positive half cycle, individuals are more mentally alert and capable. While in the latter half, thinking capacity is reduced and concentration is more difficult.

The day when the cycle changes from low to high or from high to low is a critical day. Accident studies have shown that a high percentage of accidents occur on critical days. On those days when more than one cycle crosses the baseline, a person is likely to be especially accident prone.

A biorhythm chart for a person born April 1, 1980 is shown in figure 1. Days when the curve is to the right of center are high and those to the left are low. Physical, emotional and intellectual cycles are labeled "P", "E", and "I" respectively. Critical days, zero crossings, are labeled "C". The program described in this article will provide a calendar and biorythm chart for the month you select.

Whether or not such cycles are really accurate, developing a routine to determine and plot them is an interesting AIM BASIC programming exercise. I was encouraged to do such an exercise when Mel Evans' Perpetual Calendar (MICRO April 1980, 23:27) proved such a hit with my family.

To chart the cycles, the time between birth and chart date is needed. A BASIC program to do this appeared in BYTE ("Day of The Week and Elapsed Time Programs", W. B. Agocs, September, 1979, p. 126). This program readily translated into AIM BASIC, but its 12 × 31 matrix occupied too much memory. Further, it produced an incorrect result, 365 or 366 days too large, when computing for dates in adjacent years.

It was possible to ues most of this program by substituting some simple arithmetic for the matrix. The error for adjacent years developed from the way the number of days was computed. For years between birth and chart year, 365 or 366 days were added to the sum of days in a loop which checked for leap year. For adjacent years event though there are no intervening years, the loop cycled once, at least in AIM BASIC, adding an extra year's days. In the program presented in this article, the difficulty is corrected (in step 401) by zeroing the extra days. The algorithm used in place of the matrix to find days in a year to a certain date is shown below.

Let: M = Month (a number 1 to 12)
D = Day of month (1 to 31)
S = Sum of days
MC = Correction to number of days for months of less than 31

Then: S = D +31*(M-1)-MC

In the program the correction is stored in string variable, M$ ="003344555667". The correction for the month is selected by reading the digit corresponding to the month, 0 for January and February, 3 for March and April, etc. For leap years, an additional day is added for months after February. The calculations are:

MC = VAL (MID$(M$, M, 1))

If M > 2 then S = S + L

Note: L = 1 leap year
L = 0 otherwise

```
RUN
--------------------
***BIORYTHM CHART***
--------------------
--------------------
***BIORYTHM CHART***
--------------------
FOR: MICRO BIORHYTHM

BORN:  TUESDAY

APRIL, 1 , 1980
--------------------
**** APRIL 1980 ****
 S  M  T  W  T  F  S
          1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
--------------------
    LOW    I    HIGH
----------I---------
          C
          2 *
          3  IEP
          4    IEP
          5     IEP
          6      IEP
          7      I*
          8      P*
          9     P *
         10    P EI
         11  P  EI
         12P   E I
         PC    E I
       P  14 E   I
     P      C  I
   P    E16 I
 P   E  17I
 P  E   IC
 P E   I 19
  *    I  20
 E PI    21
 E I P   22
 EI     P23
 IE      C
 I E     25 P
 I  E    26   P
 I   E   27     P
  I      E28      P
   I      C       P
    I     30 E     P
--------------------
```

FIGURE 1. BIORHYTHM FOR BIRTH APR. 1, 1980

You might want to check the calculation. Simply add the following:

```
 590 PRINT!"DAYS BET
WEEN",M1;"/";D1;"/"Y
1;"AND"
 600 PRINT!M2;"/ 1/"
;Y2
 610 GOTO10
```

Once the number of days has been computed, starting values for the biorythm sine functions are determined by subtracting the integer number of periods since birth. A natural temptation here is to compute and save all the biorythm curve values in a matrix for later plotting. Again the limited memory dictates computing, formatting and printing each day's values in sequence.

The routine to plot the cycle values assembles a twenty element printer line in a string statement, one element at a time. In this case, space eleven was chosen as the center of the plot and sine values allowed to vary nine spaces right or left. Sine values are sorted after converting to strings and adding a plotting symbol (P, E, or I). Two symbols in the same slot are plotted as an asterisk. Date is plotted in the center except at zero crossing where "C" (for critical) is plotted.

The program presented here is not as neat and pretty as it might be. Instructions for input data format, which reasonably should be included, have been omitted to save memory space. Where possible, "pirated programs " have been left intact. Some effort has been made to conserve memory space by reusing variables and using multiple statements on a line.

The program will run with print control "OFF", so that only the output information is printed. The program listing has a number of REM statements for clarity. These should be omitted when entering the program, as there is insufficient memory.

A sample Biorhythm produced by the program is shown in figure 2.

For anyone desiring further information on Biorhythms. The following publications provide a good start.

1. *Biorhythm: A Personal Science,* Bernard Gittelson, Arco Publishing Company, 2nd Ed 1976


```
--------------------
***BIORYTHM CHART***
--------------------
FOR: DAVID BURCHER

BORN:  WEDNESDAY

JULY, 9 , 1969
--------------------
**** APRIL 1980 ****
 S  M  T  W  T  F  S
          1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
--------------------
    LOW    I    HIGH
----------I---------
 I      E 1     P
 I       C   P
  I      3PE
   I    PC   E
    I 5     E
    PI      6     E
  P   I     7       E
 P        I 8         E
 P         C          E
 P        10 I        E
  P       11  I     E
   P      12    I E
     P    13    *
        P14   E  I
          C E    I
          C P      I
        E17    P    I
      E  18      P I
    E    19        *
  E      20       IP
 E       21      I P
 E       22    I P
 E       23   I  P
 E       24 I   P
  E      25I  P
   E     ICP
    E  I PC
     *   28
    *    E29
   *     C
--------------------
```

FIGURE 2. BIORHYTHM CHART PRINTED BY THE PROGRAM FOR AUTHOR'S SON.

2. *Biorhythm Sports Forecasting*, Bernard Gittelson, Arco Publishing Company, 1977

3. *Is This Your Day?*, George Thommen, Crown Publishers, Inc., Rev Ed 1973

1 REM BIORHYTHM WITH CALENDAR
5 REM USER DATE INPUTS

```
 10 L$="------------
--------"
 11 BI$="***BIORYTHM
 CHART***"
 12 PRINT!L$,BI$,L$
 15 DOT$=".........
.........":S$="*"
 35 INPUT"NAME";N$
 40 INPUT"BIRTH DATE
 M/D/Y";M1,D1,Y1
 50 INPUT"CHART DATE
 M/Y";M2,Y2
```

90 REM COMPUTE DAYS BETWEEN DATES IN SAME YEAR

```
 100 IFY2-Y1=0THEN12
0
 110 GOTO220
 120 M=M1:D=D1
 125 IFY1/4-INT(Y1/4
)=0THEN150
 130 L=0:GOSUB450
 140 GOTO160
 150 L=1:GOSUB450
 160 S1=S
 170 M=M2:D=1:GOSUB4
50
 180 S3=S-S1:GOTO580
```

Note: To conserve memory space, no input instructions have been coded. Names should be 15 digits or less. Dates are input:

Month - 2 digits
Date - 2 digits
Year - 4 digits

200 REM COMPUTE DAYS BETWEEN DATES

```
 220 S=0:M=M1:D=D1
 230 FORI=Y1+1TOY2-1
 240 IFI/4-INT(I/4)=
0THEN270
 250 S=S+365:GOTO280
 270 S=S+366
 280 NEXTI
 285 S4=S
 290 IFY1/4-INT(Y1/4
)=0THEN320
 300 L=0:GOSUB450
 310 GOTO350
 320 L=1:GOSUB450
 330 S1=366-S:GOTO36
0
 350 S1=365-S
 360 M=M2:D=1
 361 IFY2/4-INT(Y2/4
)=0THEN390
 370 L=0:GOSUB450
 380 GOTO400
 390 L=1:GOSUB450
 400 S2=S
 401 IFY2-Y1=1THENS4
=0
 410 S3=S4+S1+S2
 420 GOTO580
```

440 REM DAYS IN YEAR TO DATE SUBROUTINE

```
 450 S=D+(M-1)*31
 460 M$="00334455566
7"
 470 S=S-VAL(MID$(M$
,M,1))
 480 IFM>2THENS=S+L
 490 RETURN
```

500 REM USING ZELLERS CONGRUENCE
510 REM CALCULATE THE STARTING DAYS OF WEEK BIRTH DATE FIRST

```
 580 REM
 630 S=0
 635 Y=Y1:M=M1:D=D1
 640 IFM=1ORM=2THEN6
60
 650 GOTO705
 660 IFM=1THEN690
 670 M=12
 680 GOTO700
 690 M=11
 700 Y=Y-1:GOTO710
 705 M=M-2
 710 C=INT(Y/100+.00
5):YZ=Y-C*100
 711 D=INT(2.6*M-.1)
+D+YZ+INT(YZ/4+.1)
 715 D=D+INT(C/4+.1)
-2*C
 720 D=D-INT(D/7+.01
)*7+1
```

725 REM READ DAY AND MONTH WORDS

```
 730 FORI=1TOD:READA
$:NEXTI
 731 RESTORE:FORI=1T
OM1+7:READMN$:NEXTI:
RESTORE
 735 IFS=0THENA1$=A$
 740 S=S+1
 750 IFS=1THENY=Y2:D
=1:M=M2:GOTO640
```

825 REM COMPUTE NO. OF DAYS IN CHART MONTH

```
 826 MD$="0301010010
10":MD=31-VAL(MID$(M
D$,M2,1))
 827 IFM2=2THENMD=MD
+L
 830 R=6.2831854
```

900 REM PRINT HEADER AND CALENDAR

```
 945 PRINT" "
 960 PRINT!L$
 1000 PRINT!BI$
 1001 PRINT!L$,"FOR:
 ";N$
 1002 PRINT!TAB(20),
"BORN:  ";A1$,TAB(20
),MN$;",";D1;",";Y1
 1010 PRINT!L$
 1013 FORI=1TOM2+7:R
EADMN$:NEXTI
 1031 DS=D1
 1032 S1$=" **** ":Y
2$=STR$(Y2):MN$=MN$+
Y2$
 1033 SP=INT((20-LEN
(MN$))/2)
 1034 PRINT!RIGHT$(S
1$,SP);MN$;LEFT$(S1$
,SP)
 1035 PRINT!" S  M
T  W  T  F  S"
```

```
 1036 F$="":D1=D-.5
 1037 FORI=1TO7
 1038 DT=I-D+1
 1039 IFI<D1THENF$=F
$+" "
 1040 IFI>D1THENF$=F
$+" "+CHR$(48+DT)
 1041 IFI<6.5THENF$=
F$+" "
 1042 NEXTI
 1043 PRINT!F$
 1044 EN=0
 1045 F$=""
 1046 FORI=1TO7
 1047 DT=DT+1:IFDT>M
D+.5THENEN=1:GOTO105
3
 1048 D1=INT(DT/10+.
05):D2=DT-10*D1
 1049 IFD1<.5THENF$=
F$+" "
 1050 IFD1>.5THENF$=
F$+CHR$(48+D1)
 1051 F$=F$+CHR$(48+
D2)
 1052 IFI<6.5THENF$=
F$+" "
 1053 NEXTI:PRINT!F$
 1054 IFEN<.5THEN104
5
 1055 PRINT!L$
 1056 PRINT!"    LOW
   I    HIGH","------
----I---------"
```

1060 REM COMPUTE NO. OF DAYS SINCE LAST FULL CYCLE

```
 1061 P=S3-INT(S3/23
)*23
 1062 E=S3-INT(S3/28
)*28
 1063 C=S3-INT(S3/33
)*33
```

1064 REM PLOTTING LOOP COMPUTE P/E/I VALUES

```
 1065 FORJ=1TOMD
 1066 PP=11+INT(9*(S
IN(R*P/23)+.05))
 1067 EE=11+INT(9*(S
IN(R*E/28)+.05))
 1068 CC=11+INT(9*(S
IN(R*C/33)+.05))
```

1069 REM CHG TO STRING

```
 1070 A$(1)=STR$(PP)
+"P"
 1080 A$(2)=STR$(EE)
+"E"
 1090 A$(3)=STR$(CC)
+"I"
```

1095 REM SORT, LOW FIRST

```
 1100 F=0
 1110 FORI=1TO2
 1120 IFVAL(A$(I))<=
VAL(A$(I+1))THEN1170
 1130 T$=A$(I):A$(I)
=A$(I+1):A$(I+1)=T$
 1160 F=1
 1170 NEXTI
 1180 IFF=1THEN1100
```

1185 REM BUILD PRINT LINE

```
 1190 F$="":M=0
 1200 FORI=1TO3
 1210 A(I)=VAL(A$(I)
):A$=RIGHT$(A$(I),1)
 1220 IFA(I)=A(I-1)T
HENF$=LEFT$(F$,LEN(F
$)-1)+S$:GOTO1240
 1230 F$=F$+LEFT$(DO
T$,ABS(A(I)-A(I-1)-1
))+A$
 1240 NEXTI
 1248 F=20-LEN(F$)
 1249 IFF>0THENF$=F$
+LEFT$(DOT$,F)
 1250 I$=MID$(F$,10,
2)
 1251 IFI$="  "THENI
$=STR$(J):GOTO1290
 1252 IFRIGHT$(I$,1)
=" "THENI$="C":LC=-1
:GOTO1260
 1253 I$="C":LC=-1
 1260 F$=LEFT$(F$,9-
LC)+I$+RIGHT$(F$,9)
 1265 PRINT!F$
 1266 P=P+1:E=E+1:C=
C+1
 1270 NEXTJ
 1275 PRINT!L$,TAB(6
0)," "
 1280 GOTO1330
 1290 I$=RIGHT$(I$,L
EN(I$)-1):LC=LEN(I$)
```

1300 REM REPEAT FOR ADDITIONAL CHARTS

```
-2:GOTO1260
 1330 INPUT"MORE Y/N
";I$
 1340 IFI$<>"Y"THENE
ND
 1350 INPUT"SAME PER
SON Y/N";I$
 1360 IFI$="Y"THEND1
=DS:RESTORE:GOTO50
 1370 CLEAR:GOTO10
 1505 DATASUNDAY,MON
DAY,TUESDAY,WEDNESDA
Y
 1510 DATATHURSDAY,F
RIDAY,SATURDAY
 1515 DATAJANUARY,FE
BRUARY,MARCH,APRIL,M
AY,JUNE
 1520 DATAJULY,AUGUS
T,SEPTEMBER,OCTOBER,
NOVEMBER,DECEMBER
```

WHAT FOLLOWS IS A RUN FOR FRAN TARKENTON TO SHOW THE RESULT WITH THE "DAYS BETWEEN PRINT" ADDED. NOTE 80 BYTE FREE AFTER RUNNING.

```
NAME?
590?!"DAYS BETWEEN",
M1;"/";D1;"/"Y1;"AND
"
600?!M2;"/ 1/";Y2
610?!"FRE(0)=";FRE(0
)
RUN
--------------------
***BIORYTHM CHART***
--------------------
NAME? FRAN TARKENTON
BIRTH DATE M/D/Y? 2
?? 3
?? 1940
CHART DATE M/Y? 1
?? 1977
THERE ARE 13482
DAYS BETWEEN
 2 / 3 / 1940 AND
 1 / 1/ 1977
FRE(0)= 426
```

```
--------------------
***BIORYTHM CHART***
--------------------
FOR: FRAN TARKENTON
BORN:  SATURDAY
FEBRUARY, 3 , 1940
--------------------

*** JANUARY 1977 ***
 S  M  T  W  T  F  S
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
--------------------
     LOW    I    HIGH
---------+I---------
         I  C      P
        I E 2       P
      I E   3       P
     IE     4      P
   IE       5     P
  IE        6    P
 *          7   P
 *          8P
 *         PC
  *    P   10
    *P     11
   P*      12
 P    IE   13
 P     IE14
 P      I C
  P       C E
    P     17 I E
      P   18  I  E
        P19    I E
          C     I E
          21 P    I E
          22   P  IE
          23     P *
          24      *I
          25     E *
          26    E IP
          27   E  *
          28 E   *
           C    *
        E30   *
      E  31PI
--------------------
```

μ

Classified Ads

OSI OS 65-D UTILITY DISKETTE
Renumber, Disassembler, Memory Dump, improved CREATE, single disk Copy, disk Verify, etc. Diskette and manual $30.00, TX add 5 %. Catalog free with SASE.
GASlight Software
3820 Byron
Houston, TX 77005

SYM-1 books by Robert A. Peck Monitor Theory Manual $8.00. Hardware Theory Manual $6.00. SYM/KIM Appendix to First Book of KIM $4.25. Send SASE for details.
Datapath
P.O.Box 2231
Sunnyvale, CA 94087

Programmer Fatigue?
SYM-Bug/Monex adds 15 commands to SYM's repetoir, including an interactive trace/debug. Cassette @$200 or $3800, $15.95. 2716 (5V) $F000—$F7FF, $39.95. Commented source listing $9.95. Custom assembly add $2.00. SASE for more information.
J. Holtzman
6820 Delmar 203
St. Louis, MO 63130
(314) 863-5209

INTEGER PASCAL SYSTEM
for the Apple II. Compiler, interpreter, and translator included for $65.00. Produces 6502 code programs for high speed. 48K and Disk required. Send for free information.
M & M Software, Co.
380 N. Armando #Z-19
Anaheim, CA 92806

PROM BLUES?
1.25K Program does work for you! Verify, erase, program, copy PROM to RAM, exit. For any 6502. Loads at $0200. Listing and instructions: $9.95. Cassette: $2.00. (ASK family only) Custom assembly: $2.00. All first class PPD cont. USA. Foreign add $2.00.
Jeff Holtzman
6820 Delmar - 203
St. Louis, MO 63130

PET MACHINE LANGUAGE GUIDE
Comprehensive manual to aid the machine language programmer. More than 30 routines fully detailed. Reader can put to immediate use. NEW or OLD ROMs. $6.95 plus $.75 postage and handling. VISA/Mastercharge accepted.
Abacus Software
P.O.Box 7211
Grand Rapids, MI 49510

APPLE II SHAPE TABLE EDITOR
Makes shape table construction and editing simple. Eleven editing commands allow the user to create shapes, modify existing shapes, save and retrieve shape tables for use with the DRAW, XDRAW, and SHLOAD commands of APPLE II BASIC. Runs under firmware Applesoft floating point BASIC with 32K RAM. On cassette tape with user's manual. $9.95. Or $1.00 for manual only (refunded with purchase.
Small Systems Software
P.O. Box 40737
Washington, DC 20016

APPLE GRAPHIC AID SHEETS
LO RES 8.5" x 11" 50 sheets $2.00
HI RES 22" x 17" 20 sheets $6.00
Please add $1.00 shipping.
Boardwalk Products
225 South Western Avenue/M
Los Angeles, CA 90004

OSI TREK FANS!
Don't just WATCH Trek, PLAY it, with VIDEOTREK, the fast action graphics arcade version of Star-Trek. Klingons, Phasers, Torpedoes, Black Holes and Doomsday Machine! 8K $9.95 postpaid. Write for list of others.
Bob Retelle
2005 Whittaker Road
Ypsilanti, MI 48197

C1P/Superboard Intelligent Terminal Program transmits data from keyboard or cassette and stores received data onto cassette. Runs in 4K, selectable parity, stop bits; full/half duplex. Unique feature-user definable keyboard configuration. Lets you tailor your keyboard. $24.95 for cassette and manual. Requires RS232 mod.
Charles Shartsis
9308 Cherry Hill Rd.812
College Par, MD 20740

Continued on page 71

MICROPROCESSORS IN MEDICINE: THE 6502

MICRO is pleased to present here the first column in a continuing series by Jerry W. Froelich, M.D., on the 6502 microprocessor in medicine. The author provides his own introduction.

Soon after the introduction of commericial computers, the imaginative physician and engineer prophesied and advocated the extension of computers into medicine. The initial attempts to bring computers into the hospital only made it as far as the business office. Once computers became fully accepted in the business side of the hospital, various clinical applications were tried. One, in the clinical laboratory, was to process the vast amount of information generated by laboratory procedures. Another was to analyze electrocardiograms (EKGs), which seemed like "simple" signal analysis. Still another was monitoring, where the computer's "watchful eye" records, interprets, and reacts to deviances from preset parameters.

These applications met with various problems such as that of "technology transfer" (the man-machine interface), and size and cost limitations. One of the initial obstacles to widespread acceptance of clinical computer technology was the lack of collaboration between engineers and physicians. Sophisticated design did not translate, necessarily, into clinically practical information.

During the last decade, there has been maturation of our understanding, both of the technology transfer process and of the technology appropriate for medicine. During that same period, *microprocessors* became sufficiently economical, powerful, reliable and available to be useful and efficient for a broad range of medical applications.

Allow me to introduce myself. I first learned about computers in the late 60's by programming business applications on an IBM 1130, no small feat. The fruits of this labor financed my undergraduate education. At this time, I became interested in applying computers to the monitoring, modeling, and prediction of shock states in critically ill patients. In medical school, I began programming real-time applications in neurophysiology and nuclear cardiology. The bulk of this work was performed on the PDP 8, 11, and 12.

During my last year of medical school, I spent the summer at the National Institutes of Health (NIH) as a visiting scientist, to learn the many applications of computers to medicine at the Institutes. During this time, I helped implement real-time analysis of electrocardiograms in the post-surgical intensive care unit. It was at NIH that I first became familiar with microprocessors in medicine. Scientists there were fabricating "intelligent" monitoring devices such as preprocessors for EKGs, blood pressure measurements, and fluid ouput.

Currently, I am a clinical fellow at Harvard Medical School and work in the Radiology-Nuclear Medicine Division at the Massachusetts General Hospital in Boston. In our department, we have various applications on computers ranging from the largest to the smallest microprocessors. I myself have a PET computer, a 6800 microprocessor system (TVBUG), and access to an APPLE system.

With this introduction behind us, I would like briefly to cover what this column will accomplish. Because MICRO is dedicated to the 6502, I would like to structure the column so that an overview of a computer application in medicine includes an in-depth description of a specific 6502 application. I hope that this column can be a forum with reader comments and suggestions. What I need to know from you the reader is how much depth is desired, if reviews of current literature would be helpful, and how much technical information is desired. Please write to me at the address below.

The second column will discuss the use of microprocessors in medical education and will include a description of a specific system for teaching physicians and technologists about nuclear cardiology and nuclear medicine.

Jerry W. Froelich M.D.
9 Brown Place
Woburn, MA 01801

Hexadecimal Printer

This simple program permits the user to specify the limits within which he wants the APPLE II disassembler to operate.

LeRoy Moyer
Route 9, P.O. Box 236
Charlottesville, VA 22901

When using the disassembler in the Apple II to print out machine language code, one normally types in the starting address and then a number of L's. There are two problems with using this method to print out a machine language program. The first is that if the machine language program does not happen to be a multiple of 20 instructions, there is probably going to be a collection of unwanted garbage printed at the bottom of the desired machine code. The second problem occurs when the program being printed is fairly long. Does one type in 50 or 51 L's to get all of the desired code? The program presented here solves both of these problems by decoding an outputing the disassembled machine language code that lies between two hexidecimal addresses.

After loading the program, using it is very easy. First, turn on the printer with a control P and then type 800G (return). The screen will clear and prompt you with the header "STARTING ADDRESS". Enter the hexidecimal address of the first instruction to be decoded and then hit return. A similar prompting question will be asked for the ending address and after entering the ending address the program will start outputing the disassembled code beginning at the starting address and continuing until the ending address.

The code presented here is transportable in that only two addresses (4 bytes) need to be changed to relocate the program anywhere in memory. These locations are the addresses for the data that prints out the program's two lines of text. Data for this text is stored starting at lines 570 ($86B)

```
]A
                0010  ;DECODE BETWEEN ADR
                0020
                0030  FINA .DL 00FE
                0040  APA2 .DL 003E
                0050  LENG .DL 002F
                0060  APPC .DL 003A
                0070  ;
0800  2058FC    0080  STAR JSR $FC58    :CLEAR SCREEN
0803  A200      0090       LDX 000      :OUTPUT FIRST HEADER LINE
0805  BD6B08    0100  DBA2 LDA TIT1,X   :    "STARTING ADDRESS"
0808  F008      0110       BEQ DBA1
080A  0980      0120       ORA 080
080C  20EDFD    0130       JSR $FDED
080F  E8        0140       INX
0810  D0F3      0150       BNE DBA2
0812  206FFD    0160  DBA1 JSR $FD6F    :KEYBOARD INPUT OF STARTING ADRRESS
0815  A000      0170       LDY 000
0817  20A7FF    0180       JSR $FFA7    :CHANGE TO HEXIDECIMAL ADDRESS
081A  A53E      0190       LDA *APA2    :MOVE HEXIDECIMAL ADDRESS TO
081C  853A      0200       STA *APPC    :    APPC ($3A)
081E  A53F      0210       LDA *APA2+01
0820  853B      0220       STA *APPC+01
0822  208EFD    0230       JSR $FD8E    :PRINT LINE FEED
0825  A200      0240       LDX 000      :PRINT SECOND HEADER LINE
0827  BD7D08    0250  DBA4 LDA TIT2,X   :    "ENDING ADDRESS"
082A  F008      0260       BEQ DBA3
082C  0980      0270       ORA 080
082E  20EDFD    0280       JSR $FDED
0831  E8        0290       INX
0832  D0F3      0300       BNE DBA4
0834  206FFD    0310  DBA3 JSR $FD6F    :KEYBOARD INPUT OF ENDING ADDRESS
0837  A000      0320       LDY 000
0839  20A7FF    0330       JSR $FFA7    :CHANGE TO HEXIDECIMAL ADDRESS
083C  A53E      0340       LDA *APA2    :MOVE HEXIDECIMAL ADDRESS TO
083E  85FE      0350       STA *FINA    :    FINA ($FE) FINAL ADDRESS
0840  A53F      0360       LDA *APA2+01
0842  85FF      0370       STA *FINA+01
0844  208EFD    0380       JSR $FD8E    :PRINT LINE FEED
0847  20D0F8    0390  DBA5 JSR $F8D0    :DISASSEMBLE ONE LINE
084A  E62F      0400       INC *LENG    :INCREMENT BYTE FOR LENGTH
084C  18        0410       CLC
084D  A53A      0420       LDA *APPC    :ADD LENGTH OF INSTRUCTION TO
084F  652F      0430       ADC *LENG    :ADDRESS THAT IS POINTER FOR
0851  853A      0440       STA *APPC    :OP CODE TO BE DISASSEMBLED
0853  A53B      0450       LDA *APPC+01
0855  6900      0460       ADC 000
0857  853B      0470       STA *APPC+01
0859  38        0480       SEC
085A  A53A      0490       LDA *APPC    :SUBTRACT FINAL ADDRESS TO SEE IF
085C  E5FE      0500       SBC *FINA    :    THE END HAS BEEN REACHED
085E  A53B      0510       LDA *APPC+01
0860  E5FF      0520       SBC *FINA+01
0862  90E3      0530       BCC DBA5
0864  208EFD    0540       JSR $FD8E    :PRINT LINE FEED
0867  208EFD    0550       JSR $FD8E    :PRINT LINE FEED
086A  60        0560       RTS          :RETURN TO MONITOR
```

and 610 ($87D) in the program listing and this data is used in lines 100 ($806) and 250 ($828) respectively.

Several APPLE monitor subroutines are used in this program and two of them deserve some comment. The first is the GETNUM ($FFA7) subroutine that converts a number stored as ASCII characters in the input buffer ($200), indexed by the Y register, into a two byte hexidecimal number. This routine converts ASCII characters until it encounters a character that is a non-hexidecimal number. A carriage return ($8D) is used in this program for the terminator. The resulting hexidecimal address is stored at location A2L ($3E) and A2H ($3F) in the usual low byte, high byte order for addresses required by the 6502.

The second routine that deserves some comment is the INSTDP ($F8D0) routine. This routine disassembles an instruction and outputs it to the screen. The address that is used to direct the subroutine to the op code to be disassembled is stored in PCL ($3A) and PCH($3B). After returning from INSTDP, a number that is one less than the length of the instruction is stored in location LENGTH ($2F). The address in the pointer ($3A, $3B) is not changed by INSTDSP and hence the length of the instruction needs to be added to the pointer to get to the location of the next op code (lines 410 to 470 in the program listing).

If you do not desire to have the initial lines of text printed out on your printer then one should insert a printer turn-on routine between lines 380 and 390 of the assembled program listing. Hopefully this routine will be useful in making your machine language print-outs look neater in the future.

```
086B  535441  0570  TIT1  AS 'STARTI' :DATA FOR FIRST HEADER LINE
086E  525449
0871  4E4720  0580        AS 'NG ADD'
0874  414444
0877  524553  0590        AS 'RESS '
087A  5320
087C  00      0600        .HS 00
087D  0D      0610  TIT2  .HS 0D        :DATA FOR SECOND HEADER LINE
087E  454E44  0620        .AS 'ENDING'
0881  494E47
0884  204144  0630        .AS ' ADDRE'
0887  445245
088A  535320  0640        .AS 'SS '
088D  00      0650        .HS 00
              0660  END   .EN
```

```
SYMBOL TABLE
FINA    00FE
APA2    00DE
LENG    002F
APPC    00DA
STAR    0800
DBA2    0805
DBA1    0812
DBA4    0827
DBA3    0834
DBA5    0847
TIT1    086B
TIT2    087D
END     088E
```

μ

Programming with Pascal

This overview of PASCAL discusses the features of the language and provides a sample program illustrating its structure and ease of use.

John P. Mulligan
P.O. Box 3461
Van Nuys, CA 91407

One of the first things that I realized after purchasing my APPLE II computer system, was that programming in BASIC was really a pain. Although BASIC is very suitable for programming games and relatively simple programming systems, I feel that its usefulness declines in direct proportion to the complexity of the application. There are a number of important reasons for this.

First of all, it is very difficult to program in BASIC using Structured programming techniques. Structured programming is a concept that has become widely accepted over the last few years as a method for simplifying program design and coding, and any subsequent maintenance. Basicly, the program is designed by continuously breaking the main problem down into smaller problems, and then by writing one program module to solve each of the smaller problems. The modularization additionally serves to enhance readability and logic design.

Another aspect of programming that helps in understanding logic flow is the concept of "prettyprinting", if I may borrow a term. This is simply writing the program in such a way as to promote ease of reading, and to indicate logic flow by indentation. APPLE's BASICs are notoriously difficult to read, although this is understandable because the BASIC Interpreter needs to parse the text directly at execution time and so needs the text formatted in a specific manner.

The last, and for me, most important fault that I see with BASIC is that it is tediously slow. This again is due to the aforementioned interpretation of the BASIC textual statements. In some applications, this is quite acceptable, but for high volume processing, this becomes increasingly important. Until recently, I overcame this factor by doing most of my programming on the APPLE II in Assembly language.

Now that I have aired my grievances about BASIC, let's turn to Pascal. Pascal was first developed by Niklaus Wirth, who tried to develop the perfect programming language. This language is actually based on the ALGOL 60 programming language which is, like Pascal, a procedure oriented language. The language developed by Wirth was named after the French mathemetician Blaise Pascal, and was designed as a language to teach programming concepts. Although originally used on minicomputer systems, it has been shown to be excellent for microprocessor-based systems as well.

The nice thing about Pascal is that it has all of those traits that BASIC lacks, and more. The APPLE II implementation of U.C.S.D. Pascal is a very excellent programming system that is convenient, sophisticated and quite powerful. However, rather than concentrating on the operating system and the program development aspects of the system, I would like to talk about Pascal itself.

First of all, it is a compiler language. The program text is input to the compiler, and a Pascal P-code object module is generated that is executed by the Pascal P-machine emulation program. This speeds up program execution at least ten times over an equivalent BASIC program. Speed advanages are not the only benefit to program compilation. The program text can be written free form, which promotes the use of prettyprinting, and this in turn increases readability.

For example, I have written a program that sorts an array of integer numbers using the QUICKSORT algorithm. This is one of the most efficient sorting techniques that has been yet discovered, but it is somewhat confusing at first glance. Essentially, the array is sorted by the following means: First, the array is split into two halves and a routine is called for each half which first estimates a value that is in the middle of the range. When this is done, the array section being operated on is scanned and all values less than or greater than the estimated value are placed on their respective half of the array section. When this is accomplished, the array section is split and the procedure is again called.

Look at the program example, TESTSORT. The first thing to remember about Pascal programs is that 'first is last'. In other words, any variable, constant, or procedure must be defined before it is referred to. That is why the executable statements for any program or procedure are the last statements in that program or procedure.

A procedure is basicly a program subroutine that is, or should be accomplishing a discrete function within the program. Any procedure may also be composed of one or more procedures. In the example, procedure PRINT is a stand-alone procedure, while procedure SPLIT is constructed using four sub-modules, SWITCH, BUBBLE, MOVEUP and MOVEDN.

Notice also that variables and constants are always declared prior to their use at the beginning of the program or procedure. Additionally, these data areas are global to the lexical level of the program at which they are defined. In other words, the constant MAXMEMS is available to any statement in the program because it is defined at the highest level, but the variable HOLD can only be accessed from within procedure SWITCH.

This feature of defining variables for a sub-module allows the technique of recursion to be used. Simply put, this means that a procedure is able to call itself as a subroutine. This is in fact what the procedure SPLIT is doing. By using recursion, the programmer can keep the coding simple, and yet write extremely efficient programs. In this example, SPLIT is initially called from the main program logic, and the value 0 and the variable ACTMEMS are passed as parameters. At the end of the SPLIT processing, the size of the array segment being manipulated is evaluated. At this point, the array is broken into two halves and the procedure is called again for each half. This process continues until the array segment to be passed to the SPLIT procedure is twelve items or less. At this point, a simple bubble sort is called for efficiency reasons and the return is made from the subroutine call.

This use of recursion is possible because new and unique variables are generated for each recursional level. This allows the variables to be at the proper value when the return to the next higher level is completed. Because of this , however, a lot of memory is gobbled up in the process and there is an effective limit to the number of recursion levels possible. In the case of the APPLE, a minimum of six words are used at each level in addition to any variables used, and each word is considered by the P-machine to be 16 bits. For this reason, the example is limited to 285 members in the array.

The use of this recursion technique is what makes the QUICKSORT algorithm so efficient, however. The first sort that I wrote in Pascal was a simple bubble sort that took about 70 seconds to sort 100 items in the array. Using QUICKSORT, this same

```
 1   1   1:D     1 (*$L PRINTER:*)
 2   1   1:D     1 PROGRAM TESTSORT;
 3   1   1:D     3 (*********************************)
 4   1   1:D     3 (*                               *)
 5   1   1:D     3 (*   QUICKSORT ARRAY OF INTEGER  *)
 6   1   1:D     3 (*                               *)
 7   1   1:D     3 (*********************************)
 8  22   1:D     3 ($ }
 9  22   1:D     3
10  22   1:D     3
11  22   2:D     3     FUNCTION PADDLE(SELECT: INTEGER): INTEGER;
12  22   3:D     3     FUNCTION BUTTON(SELECT: INTEGER): BOOLEAN;
13  22   4:D     1     PROCEDURE TTLOUT(SELECT: INTEGER; DATA: BOOLEAN);
14  22   5:D     3     FUNCTION KEYPRESS: BOOLEAN;
15  22   6:D     3     FUNCTION RANDOM: INTEGER;
16  22   7:D     1     PROCEDURE RANDOMIZE;
17  22   8:D     1     PROCEDURE NOTE(PITCH,DURATION: INTEGER);
18  22   8:D     3
19  22   1:D     3   IMPLEMENTATION
20  22   1:D     1
21   1   1:D     1 USES APPLESTUFF;
22   1   1:D     3
23   1   1:D     3 CONST MAXMEMS = 284;
24   1   1:D     3
25   1   1:D     3 VAR       OUT : INTERACTIVE;
26   1   1:D   304           CON : INTERACTIVE;
27   1   1:D   605           NUM : ARRAY[0..MAXMEMS] OF INTEGER;
28   1   1:D   890    ACTMEMS,IY : INTEGER;
29   1   1:D   892        P,D,IX : INTEGER;
30   1   1:D   895       CHRCTR : CHAR;
31   1   1:D   896 (*$P*)
32   1   2:D     1 PROCEDURE PRINT(TEXT:STRING);
33   1   2:D    43 (*********************************)
34   1   2:D    43 (*                               *)
35   1   2:D    43 (*    PRINT THE INTEGER ARRAY    *)
36   1   2:D    43 (*                               *)
37   1   2:D    43 (*********************************)
38   1   2:D    43 VAR IX,CTR : INTEGER;
39   1   2:D    45
40   1   2:0     0 BEGIN
41   1   2:1     0   PAGE(OUT);
42   1   2:1    14   WRITELN(OUT,TEXT);
43   1   2:1    31   WRITELN(OUT);
44   1   2:1    38   WRITELN(OUT);
45   1   2:1    45   IX := 0;
46   1   2:1    48   CTR := 0;
47   1   2:1    51   REPEAT
48   1   2:2    51     WRITE(OUT,' ':4,NUM[IX]:6);
49   1   2:2    81     IX := IX + 1;
50   1   2:2    87     CTR := CTR + 1;
51   1   2:2    93     IF CTR = 12 THEN
52   1   2:3    99        BEGIN
53   1   2:4    99           CTR := 0;
54   1   2:4   102           WRITELN(OUT)
55   1   2:3   109        END;
56   1   2:1   109   UNTIL IX > ACTMEMS;
57   1   2:1   117   WRITELN(OUT);
58   1   2:0   124 END;
```

```
 59   1    2:0   138 (*$P*)
 60   1    3:D     1 PROCEDURE SPLIT(X,Y:INTEGER);
 61   1    3:D     3 (*******************************)
 62   1    3:D     3 (*                             *)
 63   1    3:D     3 (*   SPLIT IS A PROCEDURE WHICH   *)
 64   1    3:D     3 (*   ACTUALLY DOES THE SORTING.   *)
 65   1    3:D     3 (*   THE SORT ALGORITHM USED IS   *)
 66   1    3:D     3 (*   THE QUICKSORT METHOD.        *)
 67   1    3:D     3 (*                             *)
 68   1    3:D     3 (*******************************)
 69   1    3:D     3 VAR F,L,MID : INTEGER;
 70   1    3:D     6     ODDPASS : BOOLEAN;
 71   1    3:D     7
 72   1    4:D     1   PROCEDURE SWITCH(SW1,SW2:INTEGER);
 73   1    4:D     3   VAR HOLD : INTEGER;
 74   1    4:D     4
 75   1    4:0     0       BEGIN
 76   1    4:1     0         HOLD := NUM[SW1];
 77   1    4:1    14         NUM[SW1] := NUM[SW2];
 78   1    4:1    38         NUM[SW2] := HOLD
 79   1    4:0    49       END;
 80   1    4:0    64
 81   1    5:D     1   PROCEDURE BUBBLE(BB1,BB2:INTEGER);
 82   1    5:D     3   VAR Z,X  : INTEGER;
 83   1    5:D     5
 84   1    5:0     0       BEGIN
 85   1    5:1     0         FOR Z := BB1 TO (BB2 - 1) DO
 86   1    5:2    13         BEGIN
 87   1    5:3    13           FOR X := (Z + 1) TO BB2 DO
 88   1    5:4    26           BEGIN
 89   1    5:5    26             IF NUM[Z] > NUM[X] THEN SWITCH(Z,X);
 90   1    5:4    57           END;
 91   1    5:2    64         END;
 92   1    5:0    71       END;
 93   1    5:0    88
 94   1    6:D     1   PROCEDURE MOVEUP;
 95   1    6:0     0     BEGIN
 96   1    6:1     0       ODDPASS := FALSE;
 97   1    6:1     4       REPEAT
 98   1    6:2     4         IF NUM[F] >= NUM[L] THEN
 99   1    6:3    35           BEGIN
100   1    6:4    35             SWITCH(F,L);
101   1    6:4    43             F := F + 1;
102   1    6:4    51             MID := L;
103   1    6:4    57             EXIT(MOVEUP)
104   1    6:3    61           END
105   1    6:2    61         ELSE
106   1    6:3    63           L := L - 1;
107   1    6:1    71       UNTIL NOT (L > F);
108   1    6:0    81     END;
109   1    6:0    96
110   1    7:D     1   PROCEDURE MOVEDN;
111   1    7:0     0     BEGIN
112   1    7:1     0       ODDPASS := TRUE;
113   1    7:1     4       REPEAT
114   1    7:2     4         IF NUM[L] < NUM[F] THEN
115   1    7:3    35           BEGIN
116   1    7:4    35             SWITCH(F,L);
117   1    7:4    43             L := L - 1;
118   1    7:4    51             MID := F;
119   1    7:4    57             EXIT(MOVEDN)
120   1    7:3    61           END
121   1    7:2    61         ELSE
122   1    7:3    63           F := F + 1;
123   1    7:1    71       UNTIL NOT (L > F);
124   1    7:0    81     END;
```

array will be sorted in about five seconds. The maximum of 285 elements is sorted consistently in 16 seconds. Even though a machine language sort would run circles around these figures, try doing some sorts in BASIC. I'm not even sure that QUICKSORT could be written in BASIC.

There is one last feature of the U.C.S.D. Pascal system that I feel merits a lot of attention. With this system, machine language subroutines can be linked into and called from Pascal host programs. These routines are essentially members of Partitioned Data Sets (PDS) that are called UNITs. These UNITs each have a unique name, and up to 16 of these UNITs may reside on any one of a number of subroutine libraries that the programmer can generate. In the TESTSORT program, I wanted to use the routines NOTE and RANDOMIZE, which are machine language procedures that are used to manipulate the APPLE's speaker and in gererating random numbers, respectively. These routines reside in An APPLE supplied UNIT called APPLESTUFF. This unit is included in the program, and at the end of compilation, this UNIT is automatically linked in from the system library. Any of the functions and procedures listed at the beginning of the program above the statement, IMPLEMENTATION are now available to the Pascal host program.

In this article I have tried to highlight some of the main features of this very professional software system as simply as possible, and in doing so, have tried to indicate the usefulness of this product without being tedious. Pascal is an exciting development on the microcomputer horizon which will allow the serious software analyst to develop professiona applications for microcomputer systems. Oh yes, there is only last critical point that I have neglected to mention. Programs written in U.C.S.D. Pascal can be run on *any* computer system using the U.C.S.D. Operating System, and there are a lot of micros out there in addition to APPLE now using this operating system. Think about it for a moment. The implications are truly amazing.

```
125   1    7:0    96 (*$P*)
126   1    7:0    96 (*****************************)
127   1    7:0    96 (*                           *)
128   1    7:0    96 (*    MAIN LOGIC FOR SPLIT   *)
129   1    7:0    96 (*                           *)
130   1    7:0    96 (*****************************)
131   1    3:0     0 BEGIN
132   1    3:1     0   F := X;
133   1    3:1     3   L := Y;
134   1    3:1     6   MID := ((F + L) DIV 2);
135   1    3:1    13   IF NUM[F] < NUM[MID] THEN
136   1    3:2    40       SWITCH(F,MID);
137   1    3:1    44   IF NUM[F] > NUM[L]   THEN
138   1    3:2    71       SWITCH(F,L);
139   1    3:1    75   ODDPASS := TRUE;
140   1    3:1    78   WHILE L > F DO
141   1    3:2    83     IF ODDPASS THEN
142   1    3:3    86         MOVEUP
143   1    3:2    86     ELSE
144   1    3:3    90         MOVEDN;
145   1    3:1    94   IF (MID - X) > 12 THEN
146   1    3:2   101       SPLIT(X,MID)
147   1    3:1   103   ELSE
148   1    3:2   107       BUBBLE(X,MID);
149   1    3:1   111   IF (Y - MID) > 12 THEN
150   1    3:2   118       SPLIT(MID,Y)
151   1    3:1   120   ELSE
152   1    3:2   124       BUBBLE(MID,Y);
153   1    3:0   128 END;
154   1    3:0   142 (*$P*)
155   1    3:0   142 (*******************************)
156   1    3:0   142 (*                             *)
157   1    3:0   142 (*    PROGRAM TESTSORT LOGIC   *)
158   1    3:0   142 (*                             *)
159   1    3:0   142 (*******************************)
160   1    1:0     0 BEGIN
161   1    1:1     0   RESET(OUT,'PRINTER:');
162   1    1:1    41   RESET(CON,'CONSOLE:');
163   1    1:1    62   PAGE(CON);
164   1    1:1    72   GOTOXY(05,06); WRITE('*******************************');
165   1    1:1   120   GOTOXY(05,07); WRITE('*                             *');
166   1    1:1   168   GOTOXY(05,08); WRITE('*  INPUT NUMBER OF ELEMENTS   *');
167   1    1:1   216   GOTOXY(05,09); WRITE('*     LESS THAN 285:          *');
168   1    1:1   264   GOTOXY(05,10); WRITE('*                             *');
169   1    1:1   312   GOTOXY(05,11); WRITE('*******************************');
170   1    1:1   360   GOTOXY(26,09);
171   1    1:1   365   UNITCLEAR(1);
172   1    1:1   368   READLN(ACTMEMS);
173   1    1:1   387   RANDOMIZE;
174   1    1:1   390   FOR IY := 0 TO ACTMEMS DO NUM[IY] := (IY + RANDOM MOD 3452);
175   1    1:1   446   GOTOXY(06,13); WRITE('PRINT UNSORTED ARRAY (Y/N)? ');
176   1    1:1   491   UNITCLEAR(1);
177   1    1:1   494   READ(CHRCTR);
178   1    1:1   505   IF CHRCTR = 'Y' THEN
179   1    1:2   512     BEGIN
180   1    1:3   512       GOTOXY(06,14); WRITE('START PRINTER AND HIT ANY KEY ');
181   1    1:3   559       UNITCLEAR(1);
182   1    1:3   562       READ(CHRCTR);
183   1    1:3   573       GOTOXY(00,00);
184   1    1:3   578       PRINT('BEFORE THE SORT -')
185   1    1:2   598     END;
186   1    1:1   600   GOTOXY(12,16); WRITE('SORT INITIATED ');
187   1    1:1   632   P := 18;
188   1    1:1   636   D := 100;
189   1    1:1   640   NOTE(P,D);
190   1    1:1   649   SPLIT(0,ACTMEMS);
191   1    1:1   655   P := 18;
192   1    1:1   659   NOTE(P,D);
193   1    1:1   668   PAGE(CON);
194   1    1:1   678   GOTOXY(05,14); WRITE('START PRINTER AND HIT ANY KEY ');
195   1    1:1   725   UNITCLEAR(1);
196   1    1:1   728   READ(CHRCTR);
197   1    1:1   739   PRINT('AFTER THE SORT -')
198   1    1:0   758 END.
```

BEFORE THE SORT -

213	3303	2154	2406	1892	1348	248	1919	492	2580	23	3433
1786	1291	3451	1394	3244	2128	453	1139	1610	2982	317	3034
1813	2632	2593	2907	575	2310	1815	1938	1246	986	1506	2786
1160	3053	1433	286	1681	1820	1481	2394	2076	3004	519	1051
422	2612	1918	1708	715	1970	2371	3157	880	2612	3121	1445
929	2442	1161	2602	2043	711	3262	1640	2433	1151	1805	600
1781	3351	2234	2257	3526	2301	1320	922	1400	2658	423	383
112	2869	179	2360	2239	1770	2238	886	1168	1059	1167	404
3314	648	2967	670	2471	1920	2401	3420	2313	1246	1445	2854
1025	2014	2824	657	196	1574	1540	2854	1085	1261	1156	2703
2574	2909	1959	419	919	3212	3208	3258	2971	855	849	559
3527	1064	2566	1270	663	585	1333	989	1103	1201	1314	3220
2746	272	2456	1415	1062	303	900	1206	676	2903	1133	3210
3154	308	3573	2034	3173	2308	3482	2711	854	1817	3502	3390
582	553	2911	3056	1505	1845	1087	683	3003	3258	3317	1010
1209	1877	606	2338	785	2241	605	3221	2876	2665	830	2164
1563	3476	1433	1167	1542	1073	3005	1791	1477	3391	653	3043
951	1454	592	3326	1323	1421	2581	3609	1426	1214	1259	1836
3462	1592	1248	347	738	2298	2774	2458	2954	3116	991	2545
644	3243	2061	1381	1841	2171	1352	1568	398	2834	1764	3345
1750	1634	3661	3164	753	3690	1756	712	1019	1201	2603	1630
3486	1601	2211	3279	1122	531	2760	3020	348	302	522	874
2205	3427	1907	1857	2243	2691	3134	1570	2394	1725	713	2393
1199	1158	3477	1904	1177	318	1675	3354	2541			

AFTER THE SORT -

23	112	179	196	213	248	272	286	302	303	308	317
318	347	348	383	398	404	419	422	423	453	492	519
522	531	553	559	575	582	585	592	600	605	606	644
648	653	657	663	670	676	683	711	712	713	715	738
753	785	830	849	854	855	874	880	886	900	919	922
929	951	986	989	991	1010	1019	1025	1051	1059	1062	1064
1073	1085	1087	1103	1122	1133	1139	1151	1156	1158	1160	1161
1167	1167	1168	1177	1199	1201	1201	1206	1209	1214	1246	1246
1248	1259	1261	1270	1291	1314	1320	1323	1333	1348	1352	1381
1394	1400	1415	1421	1426	1433	1433	1445	1445	1454	1477	1481
1505	1506	1540	1542	1563	1566	1570	1574	1592	1601	1610	1630
1634	1640	1675	1681	1708	1725	1750	1756	1764	1770	1781	1786
1791	1805	1813	1815	1817	1820	1836	1841	1845	1857	1877	1892
1904	1907	1918	1919	1920	1938	1959	1970	2014	2034	2043	2061
2076	2128	2154	2164	2171	2205	2211	2234	2238	2239	2241	2243
2257	2298	2301	2308	2310	2313	2338	2360	2371	2393	2394	2394
2401	2406	2433	2442	2456	2458	2471	2541	2545	2566	2574	2580
2581	2593	2602	2603	2612	2612	2632	2658	2665	2691	2703	2711
2746	2760	2774	2786	2824	2834	2854	2854	2869	2876	2903	2907
2909	2911	2954	2967	2971	2982	3003	3004	3005	3020	3034	3043
3053	3056	3116	3121	3134	3154	3157	3164	3173	3208	3210	3212
3220	3221	3243	3244	3258	3258	3262	3279	3303	3314	3317	3326
3345	3351	3354	3390	3391	3420	3427	3433	3451	3462	3476	3477
3482	3486	3502	3526	3527	3573	3609	3661	3690			

μ

Legal Note: Computers and the Law

The Computer/Law Journal has just issued a "Call for Papers" for a special two-issue set entitled "Law and Information Policy" to be published in early 1981.

Papers are sought in all areas of the information process, from the legal aspect of fact-gathering, to information storage, retrieval and transmission. Topics include *inter alia*, privacy, protection of data bases, Viewdata and similar systems, transborder data flow, access to government data bases, cryptography, and the antitrust aspects of the telecommumications industry. Articles addressing the international aspects of information law and policy will be of particular interest.

Guest editors for these special issues are Dr. Jon Bing and Professor Selmer of the Norwegian Research Center for Computers and Law. Authors interested in submitting papers for these issues should either write Dr. Bing at the Norwegian Research Center for Computers and Law, Oslo University, Karl Johans Gt 37, Oslo 1, Norway, or write or telephone Michael Scott, Editor-in-Chief, Computer/Law Journal, 530 West Sixth Street - 10th Floor, Los Angeles, California 90014 (Telephone: 213- 623-3321).

SERIOUS SOFTWARE FOR YOUR APPLE!

EASYWRITER, the best-known, best-loved Word Processor by CAP'N SOFTWARE..... **$99**

EASYWRITER, the Professional Word-Processing System with 80-column Screen-Hardware by INFORMATION UNLIMITED SOFTWARE/CAP'N SOFTWARE..... **$250**

EASYMAILER, the mailing-list and record-management system. Use it to interface and merge lists with the Word-Processors above..... **$69**

WHATSIT? the Self-Indexing Query System by COMPUTER HEADWARE..... **$150**

VISICALC, the numerical-modeling and calculating tool by PERSONAL SOFTWARE..... **$150**

CCA, the complete Data Management System by PERSONAL SOFTWARE, contains mailing-list, report-generation & program-interfacing features..... **$99**

STOCK MARKET ANALYSIS by GALAXY..... **$49**

SUPER-CHECKBOOK does check reconciliation and analysis, by POWERSOFT..... **$20**

APPLE-DOC, documents your BASIC programs, by SOUTHWEST DATA SYSTEMS..... **$20**

MULTI-MESSAGE, allows you to create large, colorful, hi-res messages on multiple TV sets. Broadcast ads from shop-windows & trade-show booths. By CONNECTICUT INFO. SYSTEMS..... **$35**

+ APPLE DATA-GRAPH, plots up to 3 hi-res curves, 40 points each, on a graph with dimensioned axes. Graphs can be saved to disk & recalled instantly! by CONNECTICUT INFO. SYSTEMS..... **$25**

VERBATIM, 5¼-inch diskettes, per box of 10..... **$29**

****Either MULTI-MESSAGE or APPLE DATA-GRAPH are FREE with each $100 of any merchandise ordered.****

HARDWARE for your APPLE!

D.C. HAYES MICROMODEM..... **$300**

MICROSOFT Z-80 SOFTCARD, lets you run CP/M programs on APPLE..... **$300**

MOUNTAIN HARDWARE: APPLE CLOCK..... **$225**

ROMPLUS+..... **$169**

CALIFORNIA COMPUTER SYSTEMS: PARALLEL INTERFACE... **$119**

ASYNCHRONOUS SERIAL INTERFACE..... **$159**

SSM AIO SERIAL-PARALLEL INTERFACE works with almost all popular printers!..... **$175**

APPLE LANGUAGE SYSTEM with PASCAL..... **$397**

CONNECTICUT INFORMATION SYSTEMS, CO.
218 Huntington Road, Bridgeport, CT 06608
(203) 579-0472

MYSTERY HOUSE HI-RES ADVENTURE #1

Your APPLE computer becomes your eyes and ears as you enter a spooky old mansion in search of treasure. You are in complete control as you open cabinets, smash walls etc. Danger is ever present as you find your co-adventurers being murdered one by one. Can you find the killer before the killer finds you?

- OVER A HUNDRED HI-RES PICTURES
- YOUR GAME MAY BE SAVED FOR LATER CONTINUANCE
- RUNS ON BOTH 48K APPLE II AND APPLE II PLUS

Hi-Res Adventure #1 is available now at your local computer store and requires a disk drive. To order directly send $24.95 to:

On-Line Systems
36575 Mudge Ranch Road
Coarsegold, CA 93614
209-683-6858
VISA, MST CHG, COD, CHECK ACCEPTED

Look for Hi-Res Football coming soon

For use--

HI-RES GRAPHICS FOR THE APPLE II

PADDLE-GRAPHICS/TABLET GRAPHICS

The most powerful graphic development system available. Upper/lower case text may be drawn in any size, direction or color. Pictures may be sketched and filled in with any of 21 HI-RES colors (must be seen to believe!!) A shape may be constructed automatically from any object appearing on the HI-RES screen.

Paddle-graphics is for use with the standard game paddles distributed with your APPLE and TABLET-GRAPHICS is for use with APPLES' GRAPHICS TABLET.

Paddle and Tablet-Graphics are available now at your local computer store and require 48K Applesoft in rom and a disk drive. To order directly send $39.95 for Paddle-Graphics or $49.95 for Tablet-Graphics to:

On-Line Systems
36575 Mudge Ranch Road
Coarsegold, CA 93614
209-683-6858
VISA, MST CHG, COD, CHECK ACCEPTED

Look for Hi-Res Football coming soon

APPLE II PROGRAMS

VU #1 - VU #2 - FILES ea	$54.90
BINARY INSERT - ROSTER "	54.90
BROKEN ZILBOWS SERIES	$21.45
MATCHING PATTERNS	"
FOREIGN LANG. DRILL	"
WATER THE FLOWERS	"
MATH DRILL I & II	"
CATCH THE PIG	$16.45
TOUCH TYPING TUTOR	$21.45
(also in APPLESOFT)	"
APPLE MENU COOKBOOK	"
SHOPPING LIST	"
DATA ORGANIZER	"
BOWLING & TIC-TAC-TOE	$21.45
KNIGHTS & SOLITAIRE	"
APPLE LIGHT PEN	$27.95

Prices Incl. Postage & Handling

PROGRESSIVE SOFTWARE
P.O. BOX 273
PLY. MTG., PA 19462

PA residents add 6% Sales Tax

Cassette Label Program

Printed labels make cassette recordings easy to find. Now the computer can do the dirty work!

Dawn E. Ellis
RD8 Box 344
York, PA 17403

Did you ever have a hard time finding a tape that you thought you labled? Are you tired of hearing "Honey, did you see the program I've been working on?" Have you ever tried loading a program when you forgot whether it was Integer BASIC, APPLESOFT or Machine? If you answered any of these questions "yes" then this program might be the answer to your problems.

My husband and I have six cassette cases of fifteen tapes each that have no real system of labeling them. While hunting a program that just happened to be in the sixth case I looked in, I had an idea for this cassette cover printer program.

I sat down that evening after the children and my husband had gone to bed (that's the only time I can get the computer) and designed this program to use with our teletype.

I discovered that a line of 41 characters on our teletype is equal to the width of the clear plastic of a cassette tape case. Using this fact I wrote this program which when run creates a printout. The printout is designed to be cut and folded to fit the inside of the case so that the program names are clearly visible.

Program Description

Lines 150 to 190 store your name and address. Lines 200 to 260 store the inside instruction label. It was written for a tape that only has one program per side. This could be changed to suit your needs. Line 900 enables our TTY output routine. You will have to change this line to suit your own printer. Lines 1000 to 1150 print the inside label with the side-A and side-B instructions. lined 2000 to 2160 print the front cover. Use it as is or invent your own text graphics cover like I did on some of mine. If your printer has a graphics character set, you're really in luck. Lines 3000 to 3030 print the title strip. Lines 4000 to 4070 print your name and address. Line 4900 turns off our teletype motor. Line 4910 starts you all over again for the next cover. Lines 5000 to 5040 centers all inputs when they are printed.

I used colored paper for the cassette covers to make it easier to spot different types of programs. Maybe this program will help you keep your tapes organized until you win the lottery and buy a disk!

```
10  REM  CASSETTE LABEL PROGRAM
20  REM  BY DAWN E. ELLIS
30  REM  RD 8 BOX 344
40  REM  YORK PA   17403
50  DIM A$(9)
60 A$ = "":B$ = "!
                                !": REM
      39 SPACES
70 C$ = "":D$ = "!---------------
    ------------------------!"
100  REM  INPUT QUESTIONS
110  TEXT : PR# 0: HOME
120  PRINT  SPC( 9);"CASSETTE COV
    ER PRINTER"
130  PRINT  SPC( 13);"BY DAWN ELL
    IS"
140  PRINT : PRINT  SPC( 7);"PROG
    RAM FOR TAPES WITH ONE": PRINT
     SPC( 10);"PROGRAM ON EACH S
    IDE"
150  PRINT : PRINT "ENTER YOUR NA
    ME:": INPUT A$(0): IF  LEN (
    A$(0)) > 39 THEN 150
160  PRINT "ENTER FIRST LINE OF 3
     LINE ADDRESS:": INPUT A$(1)
    : IF  LEN (A$(1)) > 39 THEN
    160
170  PRINT "ENTER SECOND LINE OF
    3 LINE ADDRESS:": INPUT A$(2
    ): IF  LEN (A$(2)) > 39 THEN
    170
180  PRINT "ENTER THIRD LINE OF 3
     LINE ADDRESS:": INPUT A$(3)
    : IF  LEN (A$(3)) > 39 THEN
```

```
    180
190 PRINT : PRINT A$(0): PRINT A
    $(1): PRINT A$(2): PRINT A$(
    3): PRINT : INPUT "IS THIS C
    ORRECT? (Y OR N)  ";A$: IF A
    $ <  > "Y" THEN 150
200 HOME : PRINT : PRINT "ENTER
    SIDE-A PROGRAM NAME:": INPUT
    A$(4): IF  LEN (A$(4)) > 30 THEN
    200
210 PRINT "ENTER FIRST LINE OF 2
     LINE INST:": INPUT A$(5): IF
     LEN (A$(5)) > 39 THEN 210
220 PRINT "ENTER SECOND LINE OF
    2 LINE INST:": INPUT A$(6): IF
     LEN (A$(6)) > 39 THEN 220
230 PRINT : PRINT "ENTER SIDE-B
    PROGRAM NAME:": INPUT A$(7):
     IF  LEN (A$(7)) > 30 THEN 2
    30
240 PRINT "ENTER FIRST LINE OF 2
     LINE INST:": INPUT A$(8): IF
     LEN (A$(8)) > 39 THEN 240
250 PRINT "ENTER SECOND LINE OF
    2 LINE INST:": INPUT A$(9): IF
     LEN (A$(9)) > 39 THEN 250
260 PRINT : PRINT "SIDE-A = ";A$
    (4): PRINT A$(5): PRINT A$(6
    ): PRINT : PRINT "SIDE-B = "
    ;A$(7): PRINT A$(8): PRINT A
    $(9): PRINT : INPUT "IS THIS
     CORRECT? (Y OR N)  ";A$: IF
    A$ <  > "Y" THEN 200
900 CALL 768: PRINT : POKE  - 16
    293,0: FOR Z = 1 TO 2000: NEXT
    Z: PRINT : PRINT
990 REM  INSTRUCTION PAGE
1000  PRINT D$;" CUT"
1010  PRINT B$
1020  PRINT B$;" INSTRUCTION"
1030  PRINT B$;" PAGE"
1040  PRINT B$
1050  PRINT B$
1060 A$ = "(SIDE-A)": GOSUB 5000
1070 A$ = A$(4): GOSUB 5000
1080 A$ = A$(5): GOSUB 5000
1090 A$ = A$(6): GOSUB 5000
1100  PRINT B$
1110 A$ = "(SIDE-B)": GOSUB 5000
1120 A$ = A$(7): GOSUB 5000
1130 A$ = A$(8): GOSUB 5000
1140 A$ = A$(9): GOSUB 5000
1150  PRINT D$;" FOLD"
2000  REM COVER PAGE
2010  PRINT "!     A    PPPP   PP
    PP   L      EEEEE   !"
2020  PRINT "!    A A   P   P  P
     P  L      E       ! FRONT"
2030  PRINT "!   A   A  P   P  P
     P  L      E       ! COVER"
2040  PRINT "!   A   A  PPPP   PP
   PP   L      EEE     !"
2050  PRINT "!   AAAAA  P      P
     L      E       !"
2060  PRINT "!   A   A  P      P
     L      E       !"
2070  PRINT "!   A   A  P      P
     LLLLL  EEEEE   !"
2080  PRINT "!                   !"
2090  PRINT "!                IIIII
    IIIII              !"
2100  PRINT "!                  I
     I                 !"
2110  PRINT "!                  I
     I                 !"
2120  PRINT "!                  I
     I                 !"
2130  PRINT "!                  I
     I                 !"
2140  PRINT "!                  I
     I                 !"
2150  PRINT "!                IIIII
    IIIII              !"
2160  PRINT D$;" FOLD"
3000  REM  TITLE STRIP
3010 A$ = "SIDE-A = " + A$(4): GOSUB
    5000
3020 A$ = "SIDE-B = " + A$(7): GOSUB
    5000
3030  PRINT D$;" FOLD"
4000  REM  YOUR NAME AND ADDRESS
4010 A$ = A$(0): GOSUB 5000
4020 A$ = A$(1): GOSUB 5000
4030 A$ = A$(2): GOSUB 5000
4040 A$ = A$(3): GOSUB 5000
4050  PRINT B$
4060  PRINT D$;" CUT"
4070  PRINT : PRINT : PRINT : PRINT

4900  PR# 0: POKE  - 16294,0
4910  GOTO 200
5000  REM  CENTERING ROUTINE
5010 Z = 39 -  LEN (A$):Y =  INT
    (Z / 2):X = Z - Y
5020 C$ =  LEFT$ (B$,Y + 1) + A$ +
     RIGHT$ (B$,X + 1)
5030  PRINT C$
5040  RETURN
```

```
!--------------------------------------! CUT
!                                      !
!                                      ! INSTRUCTION
!                                      ! PAGE
!                                      !
!                                      !
!               (SIDE-A)               !
!        CASSETTE LABEL PROGRAM        !
!    APPLESOFT AND PRINTER REQUIRED    !
!                ]LOAD                 !
!                                      !
!               (SIDE-B)               !
!           MAGAZINE ARTICLE           !
!    APPLESOFT AND PRINTER REQUIRED    !
!                ]LOAD                 !
!--------------------------------------! FOLD
!     A     PPPP    PPPP    L      EEEEE !
!    A A    P   P   P   P   L      E     ! FRONT
!   A   A   P   P   P   P   L      E     ! COVER
!   A   A   PPPP    PPPP    L      EEE   !
!   AAAAA   P       P       L      E     !
!   A   A   P       P       L      E     !
!   A   A   P       P       LLLLL  EEEEE !
!                                      !
!             IIIII IIIII              !
!               I     I                !
!               I     I                !
!               I     I                !
!               I     I                !
!               I     I                !
!             IIIII IIIII              !
!--------------------------------------! FOLD
!   SIDE-A = CASSETTE LABEL PROGRAM    !
!      SIDE-B = MAGAZINE ARTICLE       !
!--------------------------------------! FOLD
!            DAWN E. ELLIS             !
!            RD 8 BOX 344              !
!                YORK                  !
!             PA    17403              !
!                                      !
!--------------------------------------! CUT
```

```
!--------------------------------------!
!                            +-------+!
! C A R D                    ! *   * !!
!                            !       !!
!  S H A R K             +-------+   !!
!                        ! *   * !*  !!
!                        !       !--+!
!                BY      !       !   !
!********     KEN & DAWN ! *   * !   !
!* A     *       ELLIS   +-------+   !
!*       *                           !
!*  /\   *                           !
!*  \/   *                H I G H    !
!*       *                           !
!*     A *             R O L L E R   !
!********                            !
!--------------------------------------!
```

```
!--------------------------------------!
!                          ****         !
!                         *COIN*        !
!   ***************************         !
!  **<><><><><><><><><><><><><>**   @   !
!  *                            *  @@@  !
!  * ********  ********  ******** * @@@ !
!  * >CHERRY< >CHERRY< >ORANGE< * @@@   !
!  * ********  ********  ******** * @@@ !
!  ****************************** @@@   !
!  *    * ONE ARM BANDIT *      *=@@@   !
!  ****************************** @@@   !
!  *          $$$$$$$$$$        * @@@   !
!  *          $JACK-POT$        *  @    !
!  *          $$$$$$$$$$        *       !
! *******************************       !
!--------------------------------------!
```

μ

ACTION, STRATEGY, AND FANTASY
for the SERIOUS games player and his APPLE II

Brain Games - 1 demands ingenuity.

Two players bombard radioactive material with protons and electrons until it reaches critical mass and sets up a ***Nuclear Reaction.*** ***Dodgem*** *requires you to outmaneuver another player to get your pieces across the board first.* ***Dueling Digits*** *and* ***Parrot*** *challenges your ability to replicate number and letter sequences.* ***Tones*** *lets you make music with your Apple (16K) CS-4004 $7.95.* ***Strategy Games*** *and* ***Brain Games*** *are on one disk (16K) CS-4503 $14.95.*

THINK

Strategy Games - 1 keeps games players in suspense.

You and your opponent trail around the screen at a quickening pace attempting to trap each other in your ***Blockade.*** *A 7 category quiz game will certify you as a* ***Genius*** *(or an errant knave!). Beginners will meet their master in* ***Checkers.*** ***Skunk*** *and* ***UFO*** *complete this classic collection (16K) CS-4003 $7.95*

Know Yourself through these valid self-tests.

Find out how your life style effects your ***Life Expectancy*** *or explore the effects of* ***Alcohol*** *on your behavior.* ***Sex Role*** *helps you to examine your behavior and attitudes in light of society's concept of sex roles.* ***Psychotherapy*** *compares your feelings, actions, and phobias to the population's norms and* ***Computer Literacy*** *tests your microcomputer savvy. A fun and instructional package (16K) CS-4301 $7.95.* ***Know Yourself*** *and* ***CAI Programs*** *are on one disk (16K) CS-4503 for $14.95*

IMAGINE

You're in command in *Space Games - 1.*

Maneuver the TIE fighters into your blaster sights and zap them with your lasers to save the rebel base camp from annihilation in ***Star Wars.*** ***Rocket Pilot*** *is an advanced real time take off and landing game. High resolution graphics, exploding saucers and sound effects add to the suspense as you repel the* ***Saucer Invasion.*** *Finally, a bonus graphics demonstration,* ***Dynamic Bouncer*** *(16K) CS-4001 $7.95.* ***Space Games*** *and* ***Sports Games*** *are on one disk (16K) CS-4501 for $14.95*

ACTION

Sports Games - 1 puts you in the Apple World Series

Take the field in the ***Great American Computer Game.*** *Mix up your pitches to keep the batter off balance. Move your fielders to snag the ball before he gets to first. Balls and strikes, double plays, force outs, and errors let you play with a realistic strategy. Also in the line up—* ***Slalom,*** *a championship downhill ski race,* ***Torpedo Alley,*** *and* ***Darts*** *(16K) CS-4002 $7.95.* ***Space Games*** *and* ***Sports Games*** *are on one disk (16K) CS-4501 for $14.95*

It's easy to order *SENSATIONAL SOFTWARE* for your Apple II.

Send payment plus $1.00 shipping and handling in the U.S. ($2.00 foreign) to Creative Computing Software, P.O. Box 789-M, Morristown, N.J. 07960. N.J. residents add $1.00 sales tax. Visa, Master Charge and American Express orders may be called in toll free to **800-631-8112** (in N.J. 201-540-0445).

Dealer Update

Once again MICRO presents a list of dealers for reference. This list is in zip-code order within the U.S., followed by Canada and Europe. Only those dealers who responded to our request for information or those who have been our dealers for several months are presented. If you now carry MICRO and are not listed, you will be included in the next update.

U.S. (zip-coded)

Programmatics
71 Sargent Avenue
Providence, RI 02906
6502: PET

American Business Computers
454 Thames Street
Groton, CT 06340
(203) 445-5166
Contact: Paul Simard, Ralph Edwards
6502: Ohio Scientific

Soft CTRL Systems
Box 599
West Milford, NJ 07480
(201) 728-1272
6502: Software and Hardware for the APPLE.

Stonehenge Computer Shop
89 Summit Avenue
Summit, NJ 07901
(201) 277-1020
Contact: Mike Mahoney
6502: APPLE, PET, Hardware

Polks/Aristo
314 5th Avenue
New York, New York 10001
(212) 279-9034
Contact: Lewis Polk

Intechnology (OSI)
23 East 20th Street
New York, New York 10003
(212) 673-6310
Contact: Hal
6502: Ohio Scientific, WP6502, Structured Program Designers

JINI Micro Systems, Inc.
Box 274
Bronx, New York 10463
(212) 796-6200
Contact: Jim Iscaro
6502: Software/JINSAM Data Manager

Designers & Builders of
Information Systems, Inc.
One Mayfair Road
Eastchester, New York 10707
(914) 779-5292
Contact: Lee Kupersmith
6502: Ohio Scientific, peripherals

Berliner Computer Center
102 Jericho Turnpike
New Hyde Park, New York 11040
(516) 775-4700
Contact: Bob Berliner
6502: APPLE dealer/service center, custom software department

Computerland of Nassau
79 Westbury Avenue
Carle Place, New York 11514
(516) 742-22262
Contact: K. Bub

Mr. Computer
Imperial Plaza, Route 9
Wappingers Falls, New York 12590
(914) 297-1223

Future Distribution, Inc.
Trimex Building, Route 11
Mooers, New York 12958
(514) 861-4741
Contact: P.H. Faure

Upstate Computer Shop
629 French Road, Campus Plaza
New Hartford, New York 13413
Contact: Tony Violante
6502: APPLE, Commodore, Atari, North Star, Printers, Terminal Furniture, Software, MICRO, Supplies and many services

Computerland/Paoli
81 E. Lancaster Avenue
Paoli, PA 19301
(215) 296-0210
Contact: D. Reece

Personal Computer
24-26 W. Lancaster Avenue
Paoli, PA 19301
(215) 647-8463
Contact: Bob Bryant
6502: Atari, APPLE

The Program Store
4200 Wisconsin Avenue NW
Washington, DC 20016
Contact: Mr. Daly

Computerland of Tysons Corner
8411 Old Court House Road
Vienna, Virginia 22180
(703) 893-0424

A I Personal/Adventure
178 Oxford Road
Fern Park, Florida 32730
Contact: Alexis Adams

Turnkey Mini-Computer Systems, Inc.
7372 NW 5th Street
Plantation, Florida 33317
(305) 791-4578
Contact: Dan Pincu
6502: APPLE and peripherals

Computerland
3020 University Drive NW
Huntsville, Alabama 35805
(205) 539-1200

Computerlab
627 S. Mendenhall
Memphis, Tennessee 38117
(901) 761-4743
Contact: James Watter
6502: Hardware, Software, books, peripherals, repair

Computerland
4579 Great Northern Boulevard
N. Olmsted, Ohio 44070
(216) 777-1433
6502: Commodore, APPLE, Synertek

Computerland
1288 Som Center Road
Mayfield Heights, Ohio 44124
(216) 461-1200
6502: Commodore, APPLE, Synertek

Computer Solutions
1932 Brown Street
Dayton, Ohio 45409
(513) 223-2348
Contact: Harvey Curran
6502: APPLE, books, magazines

Computerland Merrillville
19 West 80 Place
Merrillville, Indiana 46410
(219) 769-8020
Contact: Andy, Chuck, Debbie
6502: APPLE, PET, Atari

Digital Technology
10 N. Third Street
Lafayette, Indiana 47901
(317) 423-2548
Contact: Greg Madder
6502: APPLE

New Dimensions in Computing
541 E. Grand River
East Lansing, Michigan 48823
(517) 337-2880
Contact: Robert Gibbs

Coloma Computer Company
190 Paw Paw Street
Coloma, Michigan 49038
(616) 468-4145
Contact: Joseph Johnson
6502: Atari hardware and software

Abacus Software
P.O.Box 7211
Grand Rapids, Michigan 49510
Contact: Arnie Lee
6502: PET Machine Language guide, Tiny Pascal for PET and APPLE

Home Computer Center
2115 East 62nd Street
Indianapolis, Indiana 46220
Contact: Sandra McGee

Cyberia, Inc.
2330 Lincolnway
Ames, Iowa 50010
(515) 292-7634
6502: APPLE, Commodore, Sales and Service

Computer Emporium
3711 Douglas
Des Moines, Iowa 50310

etc. Personal Computing
6617 27th Avenue
Kenosha, Wisconsin 53140

Faragher Associates, Inc.
7635 Bluemound
Milwaukee, Wisconsin 53213
(414) 258-2588
6502: Ohio Scientific

Byte Shop
6019 W. Layton
Greenfield, Wisconsin 53220

ComputerLand
1500 South Lake Street
Mundelein, Illinois 60060
(312) 949-1300
Contact: Ted Essex

Oak Brook Computer Centre
17 W 426 22nd Street
Oakbrook Terrace, Illinois 60181
(312) 941-9005
Contact: Bill Colsher
6502: APPLE and compatible hardware and software

Data Domain of Schaumburg
1612 E. Algonquin Road
Schaumburg, Illinois 60195
(312) 397-8700
Contact: Steve Shendelman
6502: APPLE products

ComputerLand
136 Ogden Avenue
Downers Grove, Illinois 60515
(312) 964-7762
6502: APPLE, Mountain Computer, DC Hayes, Level I Service

The Computer Store
3515 Auburn Street
Rockford, Illinois 61103
(312) 962-7580
Contact: Clyde Person
6502: APPLE and Ohio Scientific

Illinois Computer Mart, Inc.
1114 W. Main Street
Carbondale, Illinois 62901
(618) 529-BYTE
Contact: Craig Martin
6502: APPLE II sales, service, programs

Computer Country
235 Dunn Road
Florissant, Montana 63031
(314) 921-4434
Contact: Jim Carroll
6502: PET, KIM, APPLE, software

Computer Country
808 Olive Street
St. Louis, Montana 63101
(314) 231-1101
Contact: Charles Tutt
6502: PET, KIM, APPLE, software

Computer Country
4479 Lemay Ferry Road
St. Louis, Montana 63129
(314) 487-2033
Contact: Greg Kirkpatrick
6502: PET, KIM, APPLE, software

Computers ASP, Inc.
7115 NW Barry Road
Kansas City, Missouri 64152
(816) 741-8013
6502: APPLE II, accessories

Personal Computer Center
3819 W 95th Street
Overland Park, Kansas 66206
(913) 649-5942

High Technology of Wichita
1038 West Pawnee
Wichita, Kansas 67213
(913) 262-0315
Contact: Richard Bowman
6502: APPLE II sales, service, software. PET sales and software, Atari sales and software

Computers Plus
5970 Broadway Boulevard
Garland, Texas 75043
(214) 840-1383

R & L Data Systems, Inc.
190 Lomax Street
Idaho Falls, Idaho 83401
(208) 529-3785
Contact: Larry L. Brown
6502: Authorized sales and service dealer for APPLE, North Star, NEC Printers, Anadex Printers, Soroc Terminals, and a complete line of floppies, along with all related supplies.

Home Computers
1775 E. Tropicana #2
Las Vegas, Nevada 89109
(702) 736-6363
Contact: Ike Jordan
6502: Commodore CBM, APPLE, Atari, and associated peripherials

Malibu Microcomputing
23910 A De Ville Way
Malibu, California 90265

Computer Forum
14052 E. Firestone Boulevard
Santa Fe Springs, California 90670
(213) 921-2111
(714) 739-0711
Contact: Mike Nell
6502: APPLE, PET, SYN1, Software, Repair, Books, Magazines

Byte Shope
5453 E. Stearns Street
Long Beach, California 90815
(213) 579-7771
6502: APPLE, PET, CBM, and peripherals

Computer World
3808 West Verdugo
Burbank, California 91505
(213) 848-5521
Contact: Joan Haller
or Stuart Mills

Silver Spur
Electronic Communication Co.
3873 Schaefer Avenue Suite F
Chino, California 91710
(714) 627-9366

Computermart of California
315 C Diamond Bar Boulevard
Diamond Bar, California 91765
(714) 598-7505
Contact: Dave Glawson
6502: APPLE, PET, Atari, Software, Books

Computerland Castro
2272 Market Street
San Francisco, California 94114

Sunset Electronics
588 San Mateo Avenue
San Bruno, California 94066
(415) 588-9705

Sunset Electronics
2254 Taraval Street
San Francisco, California 94116
(415) 665-8330

Computerland of Sacramento
1537 Howe Avenue Suite 106
Sacramento, CA 95825
Contact: Terry E. Bradley

Small Computer Systems
3149C Waialae Avenue
Honolulu, Hawaii 96818

Camera and Computer Emporium, Ltd.
921 SW Morrison
Portland, Oregon 97205
(503) 228-5242
Contact: Harry Sweeny
6502: APPLE, Peripherals, Software

Camera and Computer Emporium, Ltd.
Holly Farm Mall
16144 S.E. McLoughlin
Milwaukie, Oregon 97222
(503) 659-9191

The Computer Connection
3100 NW Bucklin Hill Road
Silverdale, Washington 98383
(206) 692-3611
6502: APPLE authorized dealer

Ye Olde Computer Shoppe, Inc.
1301 George Washington Way
Richland, Washington 99352
(509) 946-3330

Canada

The Computer Shop
3515 18th Street S.W.
Calgary, Alta.
T2T 4T9
Canada
243-0301
6502: APPLE, PET, KIM

TJB Microsystems, Ltd.
10991 124 Street
Edmonton, Alberta
Canada T6M 0H9
Contact: Jim Nerkerson
6502: Commodore PET, APPLE

McKay Systems Corp., Ltd.
The Byte Shop
2151 Burrard Street
Vancouver, B.C. V6J 3H7
Canada
(604) 738-2181
Contact: Don McKay
6502: APPLE, Atari, North Star, T.I., Centronics, and software

Compumart
411 Roosevelt Avenue
Ottawa, Ontario
K2A 1T2
Canada
(613) 725- 3192
6502: PET, APPLE, KIM

Home Computer Centre
6101 Yonge Street
Willowdale, Ontario
M2M 3W2 Canada
(416) 222-1165

Europe

Microtronic Dataprodukter
Box 401
184 00 Akersberga
Sweden
0764/65460
Contact: G. Berglund
6502: OSI

Elincom
Oosterkade 69
9503 HR Stadskanaal
The Netherlands
Contact: J. Hovius

Marketing Trim AB
Box 10031
S-100 55 Stockholm
Erik Dahlbergsg 41 — 43
Sweden 08-61 22 04
6502: AIM-65, Compukit, Acorn, Memory Boards, Books, Magazines

Portable Microsystems, Ltd.
Forby House, 18, Market Place
Brackley, Northants
England, NN13 5SF
0280 - 702017
Contact: Mike Ayers
6502: AIM-65

Up From the Basements

There are always problems and growing pains associated with an emerging market. The Ohio Scientific market is no different. The most serious threat I see to the healthy expansion of that market is software piracy. Wholesale software piracy could be particulary damaging. If that occurs, the responsibility for the damage will fall on both end-users and dealers, and the effects will injure us all.

Because the Ohio Scientific market is oriented very much toward vertical-market applications, it is particularly vulnerable to the ill effects of software piracy. Also, since this market is in its infancy, it lacks the maturity and economic pressures that keep the CP/M market balanced.

For those of you unfamiliar with the term "vertical market," it is a vertical segment of the over-all business market that can be characterized by a particular need. For example, all automobile dealers, as part of their business, handle financing, and auto financing may be considered a vertical market within the over-all financial market which cuts across all types and sizes of businesses. A computerized finance program could appeal to a range of different-sized automobile dealers, which might other wise require special-purpose software. Ohio Scientific, with its high-performance, low-priced hardware, has succeeded very well with programmers who are interested in writing their own software for such applications.

Dealers and end-users who have been successful with a particular vertical-market package are just starting to advertise their packages in various trade journals and through mailings. Their experience will largely determine the future health and quality of independent Ohio Scientific-based software. If they find their packages are routinely pirated by others, then they are unlikely to return to the marketplace with a package of similar quality. Furthermore, if a dealer or end-user purchases a package that does not meet his expectations or, worse yet, finds that he can't get support from the vendor once he receives the package, then the dealer or end-user is going to be much more cautious about entering the marketplace again.

Software piracy is already occuring on a large scale. One of the most widely advertised, independent Ohio Scientific-compatible software packages was not written by its vendors. The software purchaser already has become somewhat wary, because of the spotty quality of software and support available from the factory and from factory-supported software vendors. If software available through independent channels fails to improve on that situation, there just won't be much of a marketplace for software.

So here we sit with a burgeoning marketplace which has the potential to support very high-quality applications software. On the other hand, the marketplace could shrink to include only very low-priced software of questionable utility, with the quality software eventually going to other machines.

As an end-user, there are several things that you can do. The first is: *refuse to copy for another user any software you may have purchased.* The second is: *insist on original copies of all software and documentation that you purchase.* Original copies will generally include some sort of license form and serial number. For your own protection, you should also insist on knowing who the authors of the software are and what sort of support you can expect from both your dealer and the original vendor of the software. Though the market is still young, there is no reason why quality software should not also have quality documentation. Too often, authors are more proficient in writing code than they are in writing the English language. However, if the marketplace demands a particular quality of documentation, you can be sure that software vendors will do their best to conform to that standard. If you ever suspect that you have received a pirated copy, you should make every attempt to contact the authors. In case your dealer fails at some point to provide the support you need, that is really the only way to protect your investment.

Steps are now being taken by vendors as well. The most significant is the formation of an independent corporation, by several of the larger distributors of Ohio Scientific hardware, to distribute software and nurture the market to maturity. Though this organization is still in its natal stages, I have high hopes that it will be a strong force working in the interests of both the software vendor and the software purchaser. It will be able both to prevent and prosecute the software pirate and also serve as a resource center for software vendors and software purchasers. In addition, the new organization will insure that high-quality software will include high-quality documentation. It will also answer questions from end-users and dealers concerning installation of the packages that it offers.

If the new software corporation is as successful as I hope, and end-users realize the value of a strong, independent software market and work to make it happen, I am convinced that Ohio Scientific computers will live up to their potential as the most cost-competitive machines available today. Excellent software is being written all across the country for these machines, for every conceivable application. With the proper encouragement and a healthy marketplace, it won't be long before we all have access to it.

Jeff Beamsley
The Software Federation, Inc.
44 University Drive
Arlington Heights, IL 60004

MICRO™

PO Box 6502
Chelmsford, Mass 01824

617-256-5515

"The BEST of MICRO Volume 1" contains all of the important material from the first six issues of **MICRO** in book form.

"The BEST of MICRO Volume 2" contains all of the important material from the second six issues [#7 to 12] of **MICRO** in book form.

"ALL of MICRO Volume 2" is all six issues of Volume 2, issues 7 to 12, at a special reduced price for a limited time while supplies last.

Back Issues:

Issues 7 to 12: .....

Issues 13 on: .....

All payments must be in US dollars.
Make checks payable to: MICRO
Foreign payments in International Money Order or cash.

Subscription: One Year = 12 issues. Circle correct category and write amount in space provided.

Surface:		
United States	$15.00	
All Other Countries	$18.00	
Air Mail:		
Central America	$27.00	
Europe/So. America	$33.00	
All Other Countries	$39.00	$ .....
"BEST of MICRO Volume 1"		
Surface	$7.00	
Air Mail	$10.00	$ .....
"BEST of MICRO Volume 2"		
Surface	$9.00	
Air Mail	$13.00	$ .....
"ALL of MICRO Volume 2"		
Surface	$9.00	
Air Mail	$13.00	$ .....

No.	Surface @ $1.75 each Air Mail @$2.75 each	=	$ .....
No.	Surface @ $2.25 each Air Mail @ $3.25 each	=	$ .....
TOTAL			$ .....

If you are a subscriber, attach label or write subscription number here: .....

Name: .....

Address: .....

.....

City: ..... State: ..... Zip: .....

Country (if not U.S.): .....

Help **MICRO** bring you the info you want by completing this short questionnaire.

Microcomputers Owned/Planning to Buy: AIM SYM KIM PET APPLE OSI Other: .....

Peripherals Owned/Planning to Buy: Memory Disk Video Printer Terminal Other: .....

Microcomputer Usage: Educational Business Personal Control Games Other: .....

Languages Used: Assembler BASIC FORTH PASCAL Other: .....

Your comments and suggestions on **MICRO**: .....

.....

Club/Group User Registration Form

Name: ..... President: .....

Location: .....

No. of Members: .....

Meeting algorithm (date, time, place): .....

.....

.....

Publications: .....

.....

Aim/Purpose of the group: .....

.....

For Current Information, Contact: .....

.....

.....

Software/Hardware Catalogue Entry

Do you have a software or hardware package you want publicized? Our Software and Hardware Catalogues offer a good opportunity to receive some free advertisement. These regular features of MICRO are provided both as a service to our readers and as a service to the 6502 industry which is working hard to develop new and better software and hardware products for the 6502 based system. There is no charge for listings in these catalogues. All that is required is that material for the listing be submitted in the listing format. All information should be included. We reserve the right to edit and/or reject any submission. We might not edit the description the same way you would, so please, be brief and specific.

Name:.....

System:.....

Memory:.....

Language:.....

Hardware:.....

Description:.....

.....

.....

.....

.....

.....

.....

Copies:.....

Price:.....

Author:.....

Available from:.....

.....

.....

Classified Ad

Classified ads provide an economical way to announce new products or sales promotions, generate product interest, enhance visibility and promote good will. MICRO clusters large format classified ads at high impact locations throughout each issue. Because classifieds represent a service to readers, MICRO must restrict each advertiser to a single, six-line insertion per issue. The nominal $10 charge reflects our preparation costs and must be prepaid.

Description:.....

.....

.....

.....

.....

.....

.....

Name:.....

Address:.....

City,State:.....Zip: .....

Other

If you are interested in Writing for MICRO, becoming a MICRO dealer or advertising in MICRO, please indicate below and the information will be mailed to you.

Dealer Information Package: ☐

Advertiser's Media Package: ☐

MICRO Writer's Guide: ☐

.....

First
Class
Stamp

MICRO™

P.O. Box 6502
Chelmsford, MA 01824

The MICRO Software Catalog: XXV

Software announcements for the 6502 based systems

Mike Rowe
P. O. Box 6502
Chelmsford, MA 01824

Name: **CRYSTAL CAT.**
System: Apple II, Apple II Plus
Memory: 32K
Language: Machine
Hardware: Disk II
Description: With this program you can type "BRUNC" instead of "CATALOG" in order to: display all file names at one in alphabetical order — fast! Allow just 2 or 3 keystrokes to run any file, show binary addresses and lengths, show hidden control characters, show free space on disk, and more, operate with 1 or 2 drives using DOS 3.1 or 3.2. The program is completely relocatable; it can be run in any free 5K block of memory.
Price: $19.95
Includes: Diskette, operating instructions
Author: **Daniel J. Hughes**
Available: **JDel LectricWare, Inc.**
P.O.Box 9140
St. Louis, MO 63117

Name: **Inventory Program**
System: Apple II, Apple II Plus
Memory: 48K (Firmware Card on Apple II)
Language: Applesoft, Assembly
Hardware: 2 disk drives, 132 column printer
Description: Maintain a complete inventory on up to 800 items. Every category included to backorder as well as LOC, Coast and QTY on order. Generates search reports, keeps a running account of what was sold YTD and much more. A must for inventory control or daily inventories.
Price: $140.00 with complete manual
Author Available: **Gary E. Haffer**
Software Technology for Computers
P.O. Box 428
Belmont, MA 02178

Name: **ON-LINE DATABASE**
System: Apple II, Apple II Plus, Language System
Memory: 48K
Language: Applesoft, Machine Language
Hardware: Disk II
Description: An extremely fast and easy to use database system. Full screen editing during data entry or updating. Holds binary file pointers in memory to reduce disk access to a minimum. Allows user definition of searches and sub-sorts based on fields and sub-fields in the data records. Automatically prints full data screens, user defined reports, and mailing labels.
Price: $100.00
Author: **Mike Dhuey**
Available: **Blue Lakes Computing**
438 N. Frances
Madison, WI 53703

Name: **R.F. Power Amplifier Design**
System: Apple II with Applesoft II firmware card or Apple II Plus, with 48K RAM
Memory: 48K
Language: Firmware Applesoft
Description: This set of programs allows the user to design R.F. Power Amplifiers through the use of Microstripline, or conventional lumped constant methods. One can select either resistive or reactive impedances for interstage or transistor matching, respectively. All standard dielectric constants for standard P.C. boards are supported. Microstripline filters, splitters/combiners, and resistive matching networks can also be designed using this program. All schematics are shown in High Resolution Graphics with labels to show component location and value.
Price: $69.95
Includes: Diskette, Description and Examples
Author: **David A. Glawson**
Available: **Computermart of California**
315C Diamond Bar Blvd
Diamond Bar, CA 91765

Name: **Contest**
System: Apple II, Apple II Plus
Memory: 32K RAM—ROM Applesoft, 48K RAM(disk) Applesoft
Language: Applesoft
Hardware: Apple II, single drive. Appleclock, printed desired but optional
Description: Designed for any amateur radio operating contest. Provides not only logging contacts with other amateur operators, but in the fast-paced contest quickly tells the operator whether he has worked a particular station before. Handles

up to 900 contacts. Operator specifies call sign, signal report and section. If he has worked a particular station before, the program will tell him immediately and place him back in beginning entry mode again. Many other features.
Copies: Just released
Price: $14.95 cassette or user provided diskette. $19.95 on author supplied diskette. Specify Applesoft ROM or RAM.
Author: **Al Jensen**
Available: **Al Jensen**
19111 First Avenue
Seattle, WA 98177

Name: **PLOT**
System: Apple II, Apple II Plus
Memory: 16K
Language: Applesoft
Hardware: Apple II with Applesoft in ROM or Apple II Plus
Description: PLOT will graph virtually any function in HIRES graphics. It features automatically scaling of the y-axis, use of built in functions, and the ability to graph up to 5 functions on one set of axes. Built in funtions include ABS, SIN, COS, TAN, INT, SGN, and many more.
Copies: Just released
Price: $9.95
Author: **Joe Verzulli**
Available: **Softpoint**
Dept. A
103 Clinton Avenue
Terryville, NY 11776

Name: **SYMBOL**
System: AIM-65
Memory: 4K or more
Language: Aim-65 Assembler
Hardware: AIM-65 Assembler, with cassett
Description: Saves paper by making assembly listings unnecessary. Allows user to find a specific symbol, step up, step down, and list table to printer/display. Also calculates total number of symbols and total RAM required for table in HEX. Uses F3 key and $E00 to $FFF.
Copies: Just released
Price: $6.00 obj only, $12.00 obj and src. Will customize program for $10.00 extra
Author: **Doug Kaynor**
Available: **Software Experience**
308 NE 24th Avenune
Portland, OR 97232

Name: **PolyMan**
System: Apple II, II Plus
Memory: 48k & ROM Applesoft
64k & Language Sys.
Language: Applesoft
Hardware: Disk II, printer optional
Description: Allows interactive manipulation of twenty polynomials of degree less than 21. Add, subtract, multiply, divide, find all real and complex roots, integrate, differentiate, evaluate at real or complex point, and graph between real limits with automatic scaling of graph and annotation of extrema. Save and recall graph files and database files on disk. Display or print coefficients or roots, input by roots or coefficients.
Copies: Just released
Price: $24.95 plus 4.5 %tax in Ohio
Includes: Diskette and manual
Author: **Robert Rennard**
Available: **SmartWare**
2281 Cobble Stone Ct
Dayton, Ohio 45431

Name: **Apple World**
System: Apple II, Plus
Memory: 48K
Language: 6502 machine code
Description: Apple World is a new 3-D hires graphics package for the Apple II computers that promises to make drawing figures in three dimensions easier than ever. Is a text-editor base and is easy for anyone to us. Includes color, text-editor input, and user oriented input (no subroutines to call!).
Copies: 500
Price: $59.95 plus $5.00 s&h
Includes: 32 page instruction manual, system disk
Author: **Paul Lutkus**
Available: **USA**
750 Third Avenue
N.Y. N.Y. 10017

Name: **The Conditioning Life Dynamic**
System: Apple II
Memory: 48K
Language: Applesoft, Machine
Hardware: Apple II, Disk II
Description: This disk adds up to a long, intense look at conditioning, motivation, positive and negative reinforcements, etc. Deals with conditioned values, effects, responses, attitudes, beliefs, and motivations. Centers on five programs. One allows you to condition a Hi-res mouse to perform up to 200 sequential responses. You'll love the games, and learn from them as well.
Copies: Many
Price: $15.95
Includes: Disk, game card
Available: **Avant-Garde Creations**
P.O. Box 30161 MCC
Eugene, OR 97403

Name: **C1 Cassette Data Base Manager**
System: OSI C1 BASIC-in-ROM
Memory: 16K or more
Language: BASIC
Hardware: None special
Description: A data base manager for cassette based systems. Good for maintaining mailing lists, article or slide lists, etc. Save and read to tape, find, sort, list, add to file, delete from file, and change file entry are all supported. High speed sorts are used.
Copies: Just released
Price: $15.00 on cassette, ppd
Author: **Mike Cohen**
Available: **Orion Software Associates**
147 Main Street
Ossining, N.Y. 10562

Name: **Data Factory 3.0**
This is a revised listing of the package as presented in the August issue.
System: Apple
Memory: 48K RAM, ROM card, Language Card
Language: Applesoft
Description: A data base file program of unique utility. It can modify one of its own existing data bases by adding or deleting fields, changing the order of fields, or changing the field lengths without having to re-enter your data. This is just one of the many time and money saving features that makes this program so powerful! Can never overload your diskette; the program informs you of your free disk space when entering data. Many features!
Price: $100.00
Includes: Disk, manual and program
Author: **William Passauer**
Available: **Andent, Inc.**
1000 North Avenue
Waukegan, IL 60085

Name: **Autodialer II**
System: APPLE II
Memory: Program -9K
Data Statements 10K
Language: Applesoft
Hardware: APPLE II, APPLE II Plus with D.C. Hayes Micromodem II
Description: Comes with over 100 phone numbers built in that can be dialed automatically by the micromodem II. Each number can have up to 14 lines of information (text). Micromodem parity, wordlength, transparency, lower case mask, and duplex set at the touch of a key. Manual dial entry too!
Price: $15.00 (diskette or tape)
Author: **Bill Hyde**
Available: **Modular Software**
P.O.Box 12883
San Antonio, TX 78212

Name: **H-EDIT 1.5**
System: APPLE II
Memory: 32K ROM Applesoft. 48K with either RAM or ROM Applesoft (specify which)
Language: Applesoft, Machine Language
Hardware: APPLE II, APPLE II Plus, Disk II, Printer optional
Description: H-Edit is a 'mini' text editor which provides the user with the ability to create, read and update moderate size text files. It is a line-oriented text file editor used to speed up certain edit functions and permit any type of character input. It will provide you with the ability to create or read those 'EXEC' files that, heretofore, were a nuisance to work with. Easy to use. Comes with instruction booklet.
Copies: Just released
Price: $16.95 Diskette, plus $1.50 p&h, first class
Includes: System diskette, and booklet
Available: **ABC Software**
2802 Claude Dove
Las Cruces, New Mexico 88001

Name: **STATISTICAL**
System: APPLE II
Memory: 48K
Language: Applesoft BASIC
Hardware: Disk drive, optional printer
Description: A comprehensive statistical analysis package for econometric work including multiple linear regression of up to seven variables - each can be lagged by up to fifteen periods. Correlation with fifteen period lag. Seasonal analysis - deseasonalising and smoothing of series. Data can be stored, amended, updated and processed in arithmetic progrm before analysis. Handles both monthly and quarterly data.
Price: $69.00 diskette
Author: **Jillian Knight**
Tripont Assoc.
Systems Consultants,
Sunderland, England
Available: **B.W. Ardin**
The White House
Hutton Gate
Guisborough,
Cleveland
England

Name: **FILEWRITER**
System: APPLE II, APPLE II Plus
Memory: 2K (plus DOS & optional RAM Applesoft)
Language: Applesoft II
Hardware: Disk II, Printer optional
Description: Allows you to write random-access disk files with a minimal effort. It has many safeguarding features sothat it is very hard to make a mistake. It allows you to enter commas into your lines. The CTRL-0 command allows you to use the options the program provides. A machine language version will soon be available.
Price: $20.00 (incl. p&h)
Includes: Diskette, documentation, instructions
Author: **S. Grimm**
Available: **The Video Stop**
23492 Belaire Ct.
Los Gatos, CA 95030

Software Catalog Entries

Entries to this catalog should be submitted in the above format. Only one listing per company, per month. Entries submitted in other formats or that are too lengthy in the description portion, will be rejected. This catalog is provided free of charge and is on a first come — first served basis. Address any inquiries to:

MICRO Software Catalog
P.O. Box 6502
Chelmsford, MA 01824

6502 Bibliography: Part XXV

Dr. William R. Dial
438 Roslyn Avenue
Akron, OH 44320

703. (cont'd) Compute II, Issue 1 (April/May 1980)

Schwartz, Daniel, "Machine Language Tapes for OSI Challengers," pgs. 52-53.
Routines to enable the C1 computers to store machine-languages on tape.

Hawkins, George W., "Songs in the Key of KIM," pgs. 52-53.
Several song tables are given for Richard Martin's "Four Part Harmony" on the KIM.

Carlson, Edward H., "Fast Tape Read/Write Programs for Your OSI," pg. 56.
A complete listing for the KC fast tape Read for the OSI computers.

704. Abacus 2, Issue 3 (March 1980)

Banks, Guil, "Fresh D.O.S.," pgs. 2-3.
This routine allows the user to replace a crashed DOS on a disk without disturbing the data on the other tracks.

Anon., "Restore LN," pgs. 3-4.
This subroutine for the APPLE does a 'restore LN' where LN is a user-supplied line number.

Anon., "Computer Equivalents Chart," pg. 5.
A chart to help with binary and hex arithmetic; APPLE listing.

Davis, James P., "Truth Tables," pg. 6.
An interesting set of relationships on logic statements.

Davis, James P., "Programs," pgs. 7-8.
A series of short routines for the APPLE.

Crossman, Craig, "Fun with Assembler; An Assembly Language Tutorial," pgs. 9-11.
A tutorial.

Anon., "System Configuration," pg. 11.
A program to figure out what cards are in your APPLE slots.

705. Dr. Dobb's Journal 5, Issue 5 (May, 1980)

Barker, Lee, "Help with OSI's CPM," pgs. 36-37.
A modified listing of the OSI ten routines that corrects a few problems.

706. Nibble 1, No. 2 (April, 1980)

Micro-Sparc Staff, "Low Resolution APPLE Shape-Writer," pgs. 7-8, 46-47.
A program to assist in APPLE graphics.

Anon., "Quick and Easy," pg. 11.
Bring those decimals into line, for the APPLE.

Anon, "Short and Sweet File Names," pg. 11.
A time and space saving routine.

Anon., "How to Build and Wire Joysticks," pgs. 12-13.
Joysticks for the APPLE, a hardware article.

707. Call — APPLE 3, No. 3 (March/April 1980)

Reynolds, Lee, "Multiple Dimension Arrays in Integer BASIC," pgs. 7-11.
Extend the utility of your Integer BASIC for the APPLE.

McVay, Ray, "Integer BASIC Post-Editor," pgs. 13-16.
An illegal statement writer for the APPLE.

Dunmire, Darrell, "Personify," pg. 17.
Personalize your disks with this routine which prints over the label 'disk volume.'

Huelsdonk, Bob, "Making BASIC Behave," pgs. 18-19.
Several useful routines for the APPLE.

Throop, Wayne, "Auto-Number for Applesoft," pgs. 20-24.
A convenient utility which can be used with Applesoft BASIC on the APPLE.

Golding, Val J., "Modifying the Program Line Editor Escape Table," pg. 24.
Define your own key functions with this routine.

Golding, Val, J., "Menu," pgs. 26-27.
Use letters to call up your menu items with this APPLE utility.

Winston, Alan B., "The Multi-Lingual APPLE," pgs. 28-29.
Three listings for Pascal users.

Pilloff, Hersch, "Stock Market Data Retrieval on The Source, pgs. 30-31.
Discussion of commands and procedures for using UPI UNISTOX on THE SOURCE with the APPLE.

Aldrich, Ron, "Ron's DOS Command Finder," pg. 31.
Program to print DOS commands and addresses on the APPLE.

Golding, Val J., "Benchmarking the Micros," pgs. 33-34.
In eight benchmark programs, the APPLE averaged faster speed than most other micros. In another case, Integer BASIC showed up a hair faster than an IBM 370/115 main-frame.

Beck, Maj. Peter M., "Shape Display Utility," pg. 39.
A useful graphics utility for the APPLE.

Gibbs, Terrell T., "Do a HPLOT of Page 2," pg. 41.
How to do HIRES animation by switching from page 1 to page 2 on the APPLE.

Flanagan, Dale, "Lower Case for APTYPE," pg. 42.
Changes to allow the APTYPE program to be used on APPLEs equipped with the Paymar chip for lower case.

Stadfeld, Paul, "A Bug in the Autostart ROM," pg. 42.
A fix for a microbe in the ROM.

708. The Apple Orchard 1, No. 1 (March/April, 1980)

Crossley, John, "Applesoft Internal Entry Points," pgs.12-18.
The initial issue of the new Apple Orchard publication of the International Apple Core contains very good articles,

many of which have been abstracted before in this bibliography. This article on the Applesoft entry points is one of several reference sources contributed by Apple Computer.

Hyde, Randall, "Connecting with the USCD BIOS," pgs. 25-33.
An APPLE Pascal utility.

Stout, Bob, "Software Development Tools for the Apple II," pg. 1.
From the Apple Barrel. The first of a series of articles dealing with tools for the Apple. This one deals with a debugger.

Crosby, Mark L., "Shaping Up with the APPLE II," pgs. 37-45.
Listings for two utilities, SHAPE DESIGNER and SHAPE ASSEMBLER. Contributed by Washington Apple Pi.

Hyde, Randall, "Converting Brand X to Work with Brand Y," pgs. 47-48.
There is BASIC and then there is BASIC, but the twain *shall* meet.

Deegan, W. Curt, "APPLE Typer," pgs. 62-65.
Two utilities, APPLE Typer and Trim Print, as well as Typer II, contributed by Apple Gram.

Hyde, Randall, "Converting Integer BASIC Programs to Assembly Language," pgs. 67-71.
A tutorial on Assembly language contributed by Applesauce.

Wurzel, Bill, "The Language System — The APPLE Grows Up," pgs. 74-77.
All about Pascal, contributed by Washington Apple Pi.

Deegan, Curt, "Hex-ASCII Memory Dump," pgs. 79-80.
A utility for the APPLE, contributed by Apple-Gram.

Rivers, Jerry, "DOS Tidbits," pgs. 83-86.
Notes on disk files for the APPLE, contributed by Apple-Gram.

Suitor, Richard F., "Disk Bootstrap without ROM," pgs. 95-96.
A utility for disk users, for the APPLE. Contributed by NEAT Notes.

Knevels, Paul, "The APPLE Rumor Mill," pg. 96.
Suggested Op-codes for a microprocessor. Contributed by Apple-Gram.

709. NW Suburban Apple User Group Newsletter 1 (July 1979)

Russ, John, "A Real-Time Clock Display for the Apple Using the Mountain Hardware Board," pg. 1.
A machine program with explanations of the clock works.

710. NW Suburban Apple User Group Nwslt 3 (August 1979)

Lyle, Guy, "Memory (And Other Things Which I Forget," pgs. 1-5.
An instructive article on the APPLE Memory.

Russ, John, "Hello There!" pg. 5.
A graphics program for the APPLE.

Alpert, Dave, "Dave's Peeks," pg. 8.
How to read or use editing controls on long catalog listings that scroll off the APPLE's screen.

711. 73 Magazine, No. 236 (May 1980)

Smith, Bill, and Williams, Rich, "Breakthrough! — A Computerized Antenna Rotator," pgs. 86-94.
A ham radio utility using the KIM-1. Hardware and listing.

712. Apple Peel 1 No. 2 (Sept. 1979)

Jenkins, Jerry, "Maxwell Character Generator," pgs. 4-5.
A useful new graphics utility for the Apple.

713. Apple Bits (April, 1980)

Wysocki, Tom, "Program Page List," pg. 4.
A machine language APPLE utility.

Kovalik, Dan, "Taking the Mystery and Magic Out of Machine Language," pgs. 7-8.
A program to allow APPLE users to select any of the available text or graphics modes by pressing only one key rather than typing in a command or doing a poke.

Martie, Ed, "Investing a Different Way," pg. 11.
Use your APPLE to improve your chances at the track — can be very practicable!

714. NW Suburban Apple User Group Newslt (Sept. 1979)

Russ, John, "Formatting Numerical Input and Output," pg. 1-3.
Dress up the numbers or dollar/cents reporting in your program.

715. NW Suburban Apple User Group Newslt (October 1979)

Hartley, Tim, "Disk Volume Modification," pg. 6.
On the APPLE, how to replace "Disk Volume XXX" with "Tim's Disk #XXX."

716. 73 Magazine No. 236 (May 1980)

Walker, Bill, "VHF Signal Diffraction," pgs. 56-57.
Let your APPLE show why your ham signal may not be the best with the highest antenna.

717. NW Suburban Apple User Group Newslt (Nov. 1979)

Tapper, Rick, "Creating a Shape Table," pg. 1.

Russ, John and Russ, Chris, "Screen Access in HiRes," pgs. 3-4.
A tutorial on graphics programming.

718. Washington Apple Pi 1 No. 10 (Nov. 1979)

Crosby, Mark L., "Kaleido-Shape," pg. 8.
A shape-drawing graphics program for the APPLE.

719. NW Suburban Apple User Group Newslt (Dec. 1979)

Allen, Earl, "Applesoft Error Handling Routines," pgs. 1-9.
Examples using the ONERR statement on the APPLE.

Anon., "Control Character Locator," pg. 9.
Here is a neat routine on the Apple to find hidden control characters.

Alpert, Dave, "Dave's Peeks," pg. 10.
Locations in the APPLE Monitor for the prompt sign, HIMEM, LOMEM, Program pointer, Speed, etc.

720. Interactive Issue No. 1 (April 1980)

Anon., "Using EPROMS in AIM 65," pg. 3.
Helpful hints for using EPROMS in the AIM.

Anon., "AIM-65 Application Notes," pg. 4.
A list of eight Application Notes published by Rockwell for the 6502/AIM 65.

Anon., "AIM 65 Symbol Table Routine, " pg. 4.
Here is a short relocatable routine that will assist in obtaining a symbol table from an assembly.

Boisvert, Conrad, "A Couple of 6522 Applications Notes," pg. 9-10.
Generating Long Timed Intervals, Generating a 1 HZ Squarewave Signal.

Editor, " TTY TIP," pg. 12.
Notes on using the AIM-65 at terminal speeds of 2400 and above.

6502	7.45	10 @ 6.95	50 @ 6.55 100 @ 6.15
6502A	8.40	10 @ 7.95	50 @ 7.35 100 @ 6.90
6520 PIA	5.15	10 @ 4.90	50 @ 4.45 100 @ 4.15
6522 VIA	6.90	10 @ 6.70	50 @ 6.15 100 @5.50
6532	7.90	10 @ 7.40	50 @ 7.00 100 @ 6.60
2114-L450		4.75	20 @ 4.45 100 @ 4.15
2114-L300		5.95	20 @ 5.45 100 @ 5.10
2716 EPROM		21.00	5 @ 19.00 10 @ 17.00
4116-200ns RAM			7.00 8 @ 6.25
6550 RAM (PET 8K)			12.70
21L02			.90
S-100 Wire Wrap			$2.85 10 @ 2.65
S-100 Solder Tail			$2.35 10 @ 2.15

A P Products
15% OFF

PET 4 Voice Music System	$ 60
Zenith Z-19	$770
KIM-1	$159
SYM-1	$209
KTM 2/80	$349
Leedex Monitor	$129
Centronics 737	**$800**

All books 15% Off

DISKS
(write for quantity prices)

SCOTCH 8" Disks	10/$31.00
SCOTCH 5¼" Disks	10/$31.50
Verbatim 5¼" Disks	10/$24.50
Diskette Storage Pages	10/$ 3.95
Disk Library Cases	8" - $2.95 5" - $2.15
BASF 5¼" Disks	10/$25.00
BASF 8" Disks	10/$27.00

CASSETTES
AGFA PE-611
in 5 screw housing
C-10 10/5.65 100/48.00
C-20 10/6.45 100/57.00
C-30 10/7.30 100/66.00
All other lengths available. Write for price list.

ATARI INTRODUCTORY SPECIAL
ATARI 400 Atari 800 and all Atari Modules 20% OFF

COMMODORE PET-CBM
Write or call for quotes
NEW 8016/32 80 Column Screen
NEW 8050 950K Dual Drive
EDUCATIONAL DISCOUNTS AVAILABLE

WRITE FOR CATALOG
Add $1.00 per prepaid order for US shipping (UPS)

A B Computers
115 E. Stump Road
Montgomeryville. PA 18936
(215) 699-5826

The Apple Shoppe

JOURNAL OF APPLE APPLICATIONS Vol. No.

EDITED BY DAVID E. SMITH

PUBLISHED BY COMPU tutor

YOU BOUGHT THE BEST! NOW LEARN TO USE IT!

AT LAST!

A magazine devoted to *Applications* as well as Technique for the Apple Computer.

THE APPLE SHOPPE WILL TEACH YOU HOW TO DO ALL THOSE FANCY THINGS ON THE APPLE. LEARN HOW OTHERS ARE USING THEIR APPLES IN THE HOME, SCHOOLS AND BUSINESSES.

CHECK THESE FEATURES:

- ✓ Feature Articles on Apple Applications
- ✓ Program of the Month—'How To' with Listings
- ✓ New Products Review—All Boards, Pascal, etc.
- ✓ Language Lab—Learn Basic, Pascal, Forth, Lisp, Pilot
- ✓ Future Projects—Participate in a new program design called "The China Syndrome"
- ✓ Graphics Workshop—Learn secrets formerly known only to 'Super Programmers'

☐ YES I want to learn how to get the most out of my Apple. Send me a one year subscription. I enclose $12.

NAME: _____

ADDRESS: _____

CITY _____ STATE _____ ZIP _____ PHONE _____

☐ NO, I already know it all, but send me a free sample of next issue.

Send check or money order to: Apple Shoppe, P.O. Box 701, Placentia, CA 92670
or call (714) 996-0441

THE FLOPPY DISK ALTERNATIVE

- transfers at 4800 baud
- 500K storage per drive
- complete hardware and software system

PET/BETA-1

Meca's digital tape provides the features of disk plus gives you more storage and costs less.

FLEXIBLE DATA MANAGEMENT . . . MICRO-PROCESSOR CONTROLLED BETA-1 UNIVERSAL TAPE DRIVE BY MECA* . . . PERIPHERAL DEVICE WITH COMPLETE SOFTWARE SUPPORT

SPEED	Loads and saves at 4800 baud. Seeks at 100"/sec
STORAGE	Each BETA-1 drive stores 525,000 bytes or 126 named programs or files on a single cassette. Expands to four drives—two megabytes
OPERATING SYSTEM	24 commands, 27 error messages, 2 user-defined commands—fully compatible with PET BASIC. Resides in only 4K—abbreviated 2K version, bootstrap loader, all romable
FILE CONTROL	Named files, directories, program chaining and merging, sequential and random access files, field and record definitions, and more . . .

LOAD AND SAVE AT HIGH SPEED

Eliminate those piles of cassettes cluttering up your life. Organized on a single cassette, your present collection of programs becomes a named program library. You load and save any program in seconds, instead of minutes. In addition to fast, convenient, reliable program storage, huge amounts of textual and numeric data can be accessed quickly and easily. With up to two megabytes on line, programs for mailing lists and business records are practical.

This is a complete system. Just plug it in and go. Additional software is available.

BETA-1 DRIVE, PET/BETA-1 OPERATING SYSTEM, MANUAL . . .	$555.00**
ADDITIONAL BETA-1 SLAVE DRIVES	310.00
MANUAL (applicable to purchase)	10.00

F S S
software for small computers

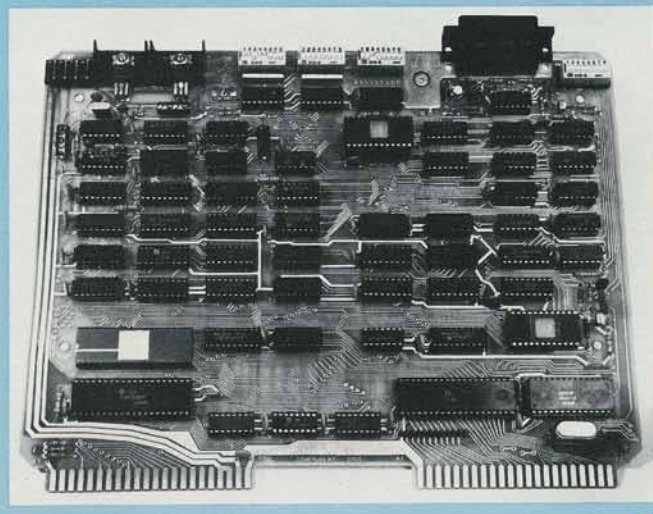
1903 Rio Grande
Austin, Texas
78705

1-512-477-2207

P.O. Box 8483
Austin, Texas
78712

* FSS IS AN AUTHORIZED MECA INC. DEALER
**TEXAS RESIDENTS ADD 5% SALES TAX
PET IS A REGISTERED TRADEMARK OF COMMODORE INC.

The ONLY 'PLUG IN AND GO' VIDEO BOARD for Both the Beginning and the Sophisticated System Builder!



FEATURES

ASK VIDEO PLUS II ™ Software EPROM
Fully AIM, SYM, KIM, Compatible. Works with Monitors, BASIC, Editors. Observes all Standard Programming Conventions.
Supports AIM keyboard, upper and lower case, or any ASCII keyboard.
Fast Scrolling and Flicker-free Operation
ASK Software fully supports all VIDEO PLUS II options.

BENEFITS

Instant Video Display Capability
Simple cable connection provides for easy installation, and immediate use: simply 'plug in and go.' Standard Options are available to meet user's requirements for future major system improvements.

VIDEO PLUS II Hardware
EPROM Character Generators provide for
128 Character Set with 2716 EPROM,
or 256 Character Set with customer furnished 2532.
Programmable Character Size
Selective Character Blank/Unblank
Improved Keyboard Interface
Reverse Video

Display requirements may be tailored to meet actual application requirements on an individual basis. User may specify character set, height, width and spacing in accordance with specific application display and man/machine requirements. 2000-4000 characters or limited, high resolution graphic screen displays.

VIDEO PLUS II Additional Capabilities
Up to 2K Display RAM, 2K Program Character Generator
6502 Stand-alone Operations
ACIA Communications Provision
- Handles RS 232 and 20MA Current TTY loop.
- Programmable baud rates, 50 to 19.2K baud.

VIDEO PLUS II supports major system enhancements which can significantly extend a product's life cycle. Any combination of options may be specified. Additional capability may be added to meet changing application requirements or planned product line improvements.

VIDEO PLUS II — Provides Unexcelled Cost/Performance

STANDARD BOARD — $295.00 Includes: ASK VIDEO Software EPROM, Character Generator EPROM (2716), RAM 1K, Configuration DIP Switches (24), and CRT Controller. Options: Additional 4K RAM — $50.00. 6502 Stand-alone Options — $20.00. Communications Option — $35.00.

DEALER and OEM QUANTITY DISCOUNTS AVAILABLE

All prices shown are US and Canadian only, and are exclusive of shipping charges and applicable taxes. Other improved products include **MOTHER PLUS II**, **PROTO PLUS II**, **A PLUS II**, and **AIM PLUS II**. Volumes shipments for all PLUS II products commence Oct. 1, 1980. For more information contact:

THE COMPUTERIST®
34 Chelmsford St., Chelmsford, MA. 01824 617/256-3649

Cost Effective Systems for the Microcomputer OEM.

Ohio Scientific has been building small business microcomputers and personal computers since the beginning of the microcomputer revolution. Most Ohio Scientific products incorporate a bus architecture utilizing modular circuit cards mated to a multi-slot backplane. Ohio Scientific's 48 signal line bus is designed to effectively marry the versatility and modularity of bus architecture with the economies of consumer products producing an ultra-low cost yet reliable system. Many industrial users of microcomputers recognize the economy and versatility of Ohio Scientific's modular computer boards and utilize these boards and subsystems as well as customers who purchase complete computer systems on an OEM basis.

Ohio Scientific's New OEM Program

Ohio Scientific now recognizes the importance of the OEM marketplace and is introducing a complete program for the board level OEM user as well as the system OEM. The program starts with our standard products including three CPU boards, a broad range of static and dynamic memory boards, mini and 8" floppy disk controllers, printer controllers, multiple RS-232 port boards, a hard disk controller, and video interface with optional keyboard. Backplanes with two, four, eight or sixteen slots are available. These standard products are now being supplemented by a broad range of products specifically for the OEM user including:

- New universal telephone interface board which has auto-dial capability, auto-answer capability, tone encoding and decoding, answer and originate 300 baud modem and voice I/O via tape recorder or optional phonetic voice output system.
- New calendar-clock with several month battery backup capability which can be programmed to automatically restart the computer or shut off the computer at set times. The circuit board also includes automatic power-fail restart capability.
- Instrumentation quality high speed, 12-bit analog A/D – D/A module with a 16-channel input multiplexer and two 12-bit D/A converters.
- A large range of parallel interface options including circuit cards containing 48 parallel I/O lines.
- New solderless prototyping board which connects to the computer system and allows rapid prototyping of new interface ideas.
- System PROM blaster which accepts 8K through 64K bit industry standard EPROMS and a universal EPROM-ROM card.
- A card edge extender, bus analyzer and bus compatible breadboards.

Documentation

All of Ohio Scientific major circuit boards are now fully documented by Howard Sams (the originator of the Sams Photofact series for Consumer Electronics) servicing manuals which include block diagrams, schematics, detailed pictorials, parts placement diagrams and parts lists providing the designer, systems integrater and serviceman with detailed hardware information. Ohio Scientific is offering qualified OEM users its principal disk operating system (OS-63D V3.2) which supports multiple languages, mini-floppies, 8" floppies, printers, modems and other accessories in documented Source Code and machine readable form which can be reassembled on standard OSI computers. This gives the product developer the ultimate flexibility in integrating these components into his total system design.

Best of all is the Price

Because of Ohio Scientific's hundreds of thousands of boards per year volume for the consumer and small business market, these products cost a mere fraction of the corresponding LSI-11, SBC or S-100 bus boards. This economy allows you to utilize a floppy subsystem in your product at a total cost typically less than an EPROM based system from other vendors.

Ohio Scientific's reasonably priced universal telephone interface and voice output capabilities allow you to integrate advanced telecommunications, remote control capabilities and/or unlimited vocabulary voice response in your systems at the same price as a "bare bones" implementation with other bus architectures.

Easy to Start With

Getting started with the OSI bus architecture is now easy with documentation, off the shelf availability, and economical computer systems for in-house software development using Assembler, BASIC FORTRAN or PASCAL. Ohio Scientific's new OEM contract provides easy to start with terms and generous volume discounts.

For more information and the name and phone number of your local Ohio Scientific OEM representative call 1-800-321-6850 toll free. Please specify your interest as an OEM user.

OHIO SCIENTIFIC
1333 SOUTH CHILLICOTHE ROAD
AURORA, OH 44202 • (216) 831-5600