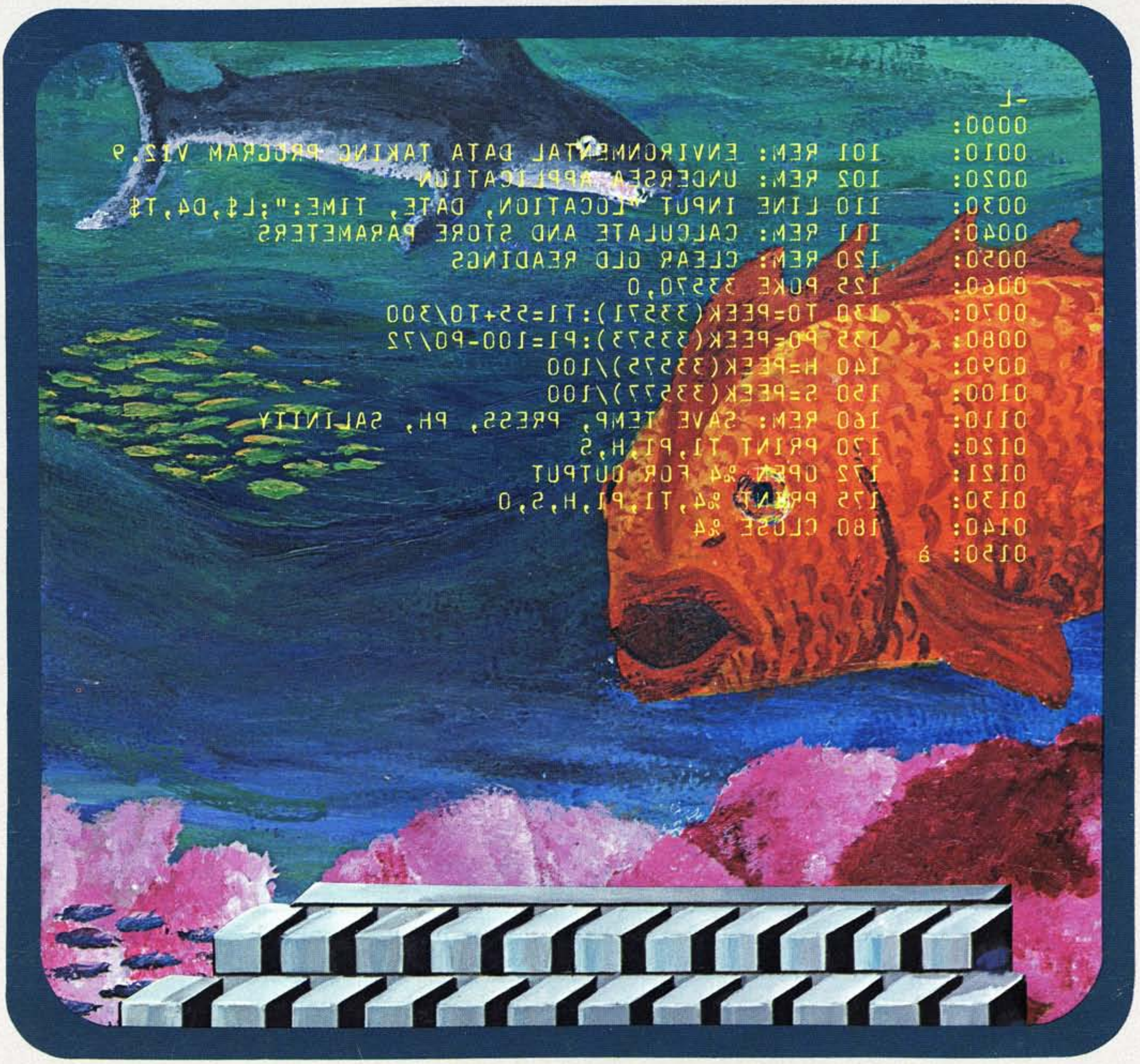
MICRO™

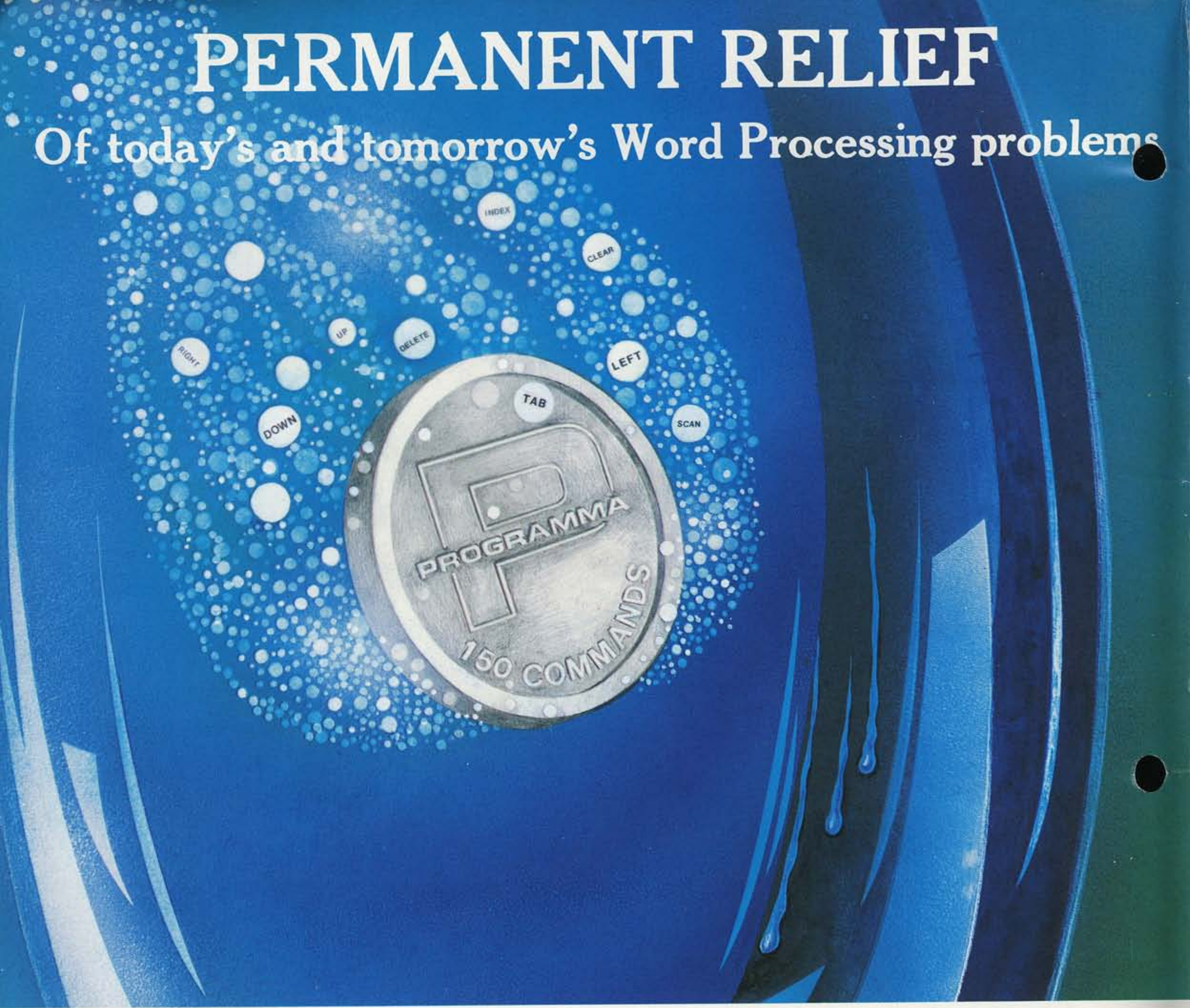
THE 6502 JOURNAL



No. 30 November 1980 $2.00

PERMANENT RELIEF

Of today's and tomorrow's Word Processing problems



Apple PIE + Formatter

Apple PIE (Programma International Editor) and FORMAT (text formatter) offer full strength solutions to today's word processing problems. These versatile, powerful programs provide document preparation and word processing capabilities previously found only on much larger computer systems.

PIE is a general purpose, full screen editor that uses control keys and function buttons to provide a full range of editing capabilities such as search and replace, delete, copy, insert, move. Changes may be made directly anywhere on the screen and are shown as they are performed.

FORMAT uses simple instructions embedded in the input text to describe the desired appearance of the final document. It handles centering, underlining, indenting, page numbering, margins, headers, footers, even form letters, and includes a proofing capability.

These high-quality, cost-effective programs come with comprehensive documentation and run on a 32K Apple II. They are available through your local computer store or direct from Programma International, Inc. at the introductory price of $79.95*.

VIDEX VERSION T.M.
DOUBLE VISION T.M.
SUPR TERM VERSION T.M.
STANDARD VERSION

*December 1, $129.95.

PROGRAMMA
3400 Wilshire Boulevard
Los Angeles, California 90010

or

Simple enough for the beginner. Versatile enough for the professional.

MICRO™

November 1980
Issue Number 30

Staff

Editor/Publisher
Robert M. Tripp

Associate Publisher
Richard Rettig

Associate Editor
Mary Ann Curtis

Director Sales/Marketing
James Anderson

Advertising Manager
L. Catherine Bland

Circulation Manager
Carol A. Stark

MICRO Specialists
APPLE: Ford Cavallari
PET: Loren Wright
OSI: Paul Geffen

Comptroller
Donna M. Tripp

ARTICLES

DEPARTMENTS and SUNDRY

MICRO™ is published monthly by:
MICRO INK, Inc., Chelmsford, MA 01824
Second Class postage paid at:
Chelmsford, MA 01824
Publication Number: COTR 395770
ISSN: 0271-9002

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

Software Distribution

This editorial is in two parts. Part I appeared last month.

Part 2

Some Basic Questions To Consider.

1. Is your program worthwhile? Just because you wrote it and think it's great does not mean others will. Show your software to knowledgeable friends, computer club members, and local dealers. Get their honest evaluations, and listen to them. They might persuade you not to bother selling your program. They might convince you that it really is worthwhile. Or, they might even give you some valuable suggestions for improving it.

2. Is your program unique? What the world needs now is *not* another checkbook balancing package. If your program is too similar to products already on the market, it will naturally reduce your chances of success. Friends, clubs, and dealers can assist in determining what is available. The major magazines often list software products (the MICRO Software Catalog for example) and carry ads from software houses. Check catalogs of the major software houses. Since you may want to have a software house distribute your material, contact several. They will be able to estimate the value of your material on the current market.

3. What will your package sell for? In addition to the procedures suggested above, check in magazines, catalogs and stores on what programs of similar complexity and size are currently going for. In figuring your production costs, remember that printing booklets and copying tapes or diskettes can get expensive in small quantities. Advertising and distribution costs must be included as well.

4. How should your product be packaged? First consider how you plan to distribute the material. Mail-order packaging can be minimal. Your customer is not buying the product because of the package. However, store packaging is very important, since your product will be competing with many others for the buyer's attention and dollars.

If all of the above questions—and the list is by no means exhaustive—cause you to have second thoughts about selling your software yourself—good! Do not rush into selling software blindly. It can be profitable, even lucrative, but it does take time, money, and effort.

Sell it as an article. If, after careful consideration, you decide that your particular software is not extremely marketable, but you still believe that it has merit and should be distributed, then how about publishing it? Most, but not all, national magazines pay for material they publish. Most editors prefer articles which include programs. You should consider a number of factors in selecting the magazine to which you submit your material. Is your program the type they normally print? Will the audience of the magazine be interested in your program? Does the magazine pay at competitive rates? Does the publisher pay residual rights, that is, if your work appears in a "Best Of..." or some other reprint form, do you get additional payments? (MICRO's policy is to make residual payments; many other publishers do not.)

If you decide to sell your software as an article, then you may want to re-evaluate your presentation. An article is generally most valuable when it can discuss and describe a technique, methodology, programming trick, or some other aspect of programming which may have value above and beyond the particular application. Your article should emphasize any unique or interesting aspects of the program in addition to presenting the basic information required to use the material. This will maximize both the chance of you article being accepted at top dollar and its usefulness to the reader.

Summary If you have a good piece of software that should be shared with others, please do not let it lie idle. If you want to spend minimal effort to get it out to others, then give it away. You can make some money on the right type of software by writing it up as an article. The greatest payoff can be in selling a software package, either directly or through a software distribution company, but that does entail additional work on your part. So, tear yourself away from your micro computer long enough to get your work distributed—at least for personal credit, and possibly for cash.

Cover Artist
Liz Jeffrey

The 6502 Microprocessor

Is there something fishy about the cover? You will probably never see a microcomputer such as the Apple, PET, etc. at the bottom of the ocean. They are not intended for such extreme environments. The basic building block of our familiar microcomputer, the 6502 microprocessor, could quite easily be found in such a situation. As we trace our ancestry back to the sea, our microcomputers have evolved from the microprocessor.

The goal of the designers of the various microprocessors such as the 8080, 6800 and 6502 was not to build microcomputers. As the name implies, these devices were intended to be sophisticated **process** controllers, not microcomputers. Many of the "limitations" of these devices can be understood when the original intent is considered. For example, addressing modes which would permit simple program relocation, a powerful tool in a general purpose computer systems, are not provided. That makes sense, however, if you consider that a process controller will normally have its program in ROM, making relocatability useless. A number of other trade-offs were made in the design, generally favoring **processing** over **computing**. The richness of the I/O capabilities vs. the lack of multiply and divide instructions is another example.

There is nothing inherently wrong with using the 6502 microprocessor in areas beyond its initial design scope. It would be nice, in view of its use as a microcomputer element, if its power for **computing** could be improved. New products are being released in the 8080 and 6800 lines. It would be nice to see some upgrading of the 6502. A number of suggestions for enhancements have been submitted by MICRO readers, and will appear in the next issue. Rockwell, Synertek, Commodore, are you listening?

How to Use the Hooks

There are a lot of great things you can do with your APPLE, once you know how to use the available hooks.

Richard Williams
4380 Albany Drive #23
San Jose, CA 95129

The APPLE II allows the user easily to substitute his own input and output routines for the standard ones. Figure 1 shows the basic flow of control when a character is output by the APPLE II. Figure 2 shows how the control path changes when the user substitutes his own output routine for the standard monitor path. By using what are known as "hooks," the user can break the normal flow of control and redirect it to his own routine.

An example of how this can be used is shown in figure 3. Control characters normally do not show on the screen. However, by inserting a routine to change control characters into inverse video when printed, the characters will show on the screen. This is very useful for listing programs containing control characters.

How It Works

Before doing the actual input or output, the system does an indirect jump, via the zero page, to the actual input or output routine. By changing the jump address, the user can substitute his own routine for the standard zone. For input, at location $FD18 in the monitor, there is a JMP (KSWL) instruction. KSWL (at $38) and KSWH (at $39) contain the address of the input routine with the low byte specified first. Similarly, at address $FDED, there is a JSR (CSWL) instruction which is the jump to the output routine. CSWL, address $36, and CSWH, at $37, contain the address of the output routine. This code can be seen on pages 85 and 86 of the red APPLE II reference manual.

How to Insert an Input Routine

The normal input routine is KEYIN at address $FD1B. To replace it with your routine, store its address in KSWL and KSWH. Your input routine needs to do the following:

1. Upon entry to your routine, the accumulator will contain the character replaced by the flashing prompt. You must restore this character on the screen by doing a STA (BASL), Y where BASL = $28. Do this before altering the A or Y registers.

2. Clear the keyboard strobe, if the character came from the keyboard.

3. Return the character, with the high bit set, in the accumulator.

4. The normal input routine increments the random number seed while it waits for input. You should do this also.

If you wish to get your input from the keyboard, you can do all of these by doing a call to KEYIN (JSR $FD1B). You can then do whatever

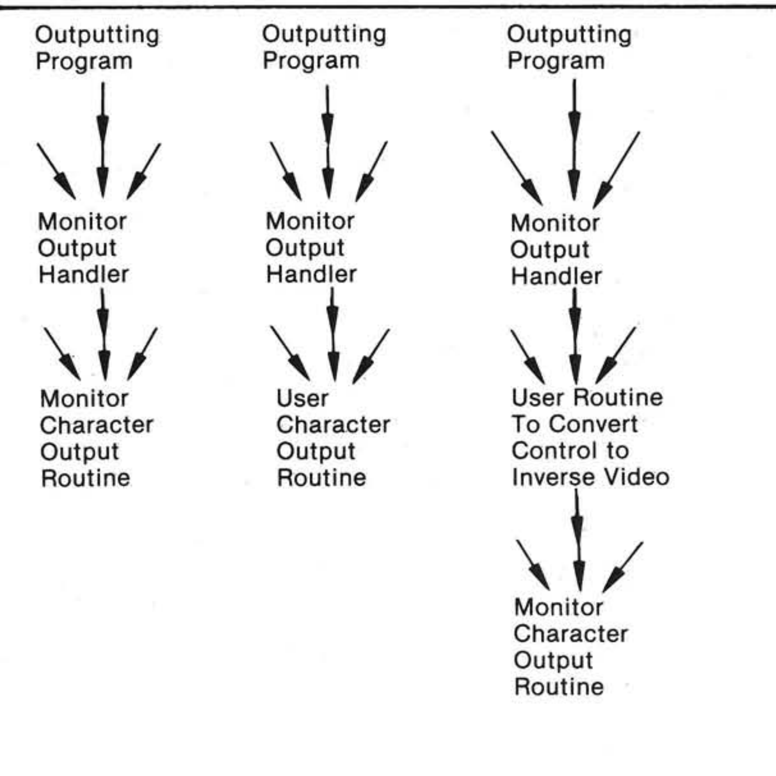


Figure 1 *Figure 2* *Figure 3*

processing that you want on the character, which is in the accumulator, and then return with an RTS. If you write your own routine to replace KEYIN, you should first carefully study KEYIN.

How to Insert an Output Routine

The normal output routine is COUT1 (address $FDF0). To insert your routine, store its address in CSWL and CSWH (addresses $36 and $37) with the low byte first. The character to be output will be placed in the accumulator before your routine is called. If you wish the character in the accumulator to be printed on the screen after you are done, exit your routine by doing a JMP COUT1. A routine to convert control characters to inverse video is an example of this.

How to Remove the Routines

The input and output routines can be removed from the hooks by typing IN#0 or PR#0 respectively. Or, if done in a program, a JSR SETKBD (address $FE89) simulates a IN#0, and a JSR SETVID (address $FE93) simulates a PR#0.

Special Notes for DOS Users

If you are using the disk operating system (DOS), you must follow some special rules when attaching or removing your routines. DOS normally sits in both the input and output hooks itself. Consequently, when you alter the hooks, you must call a DOS routine which informs DOS that the hooks have been changed. DOS will then reconnect itself to the hooks, but it will use your routines instead of the standard I/O routines. The routine to do this is at $3EA.

Example

The sample program in figure 4 inserts or removes a routine from the input hook.

```
SOURCE FILE: NEWKEYS
00DC:              1 BKSLSH  EQU  220        ;ASCII BACKSLASH
008B:              2 CTRLK   EQU  139        ;ASCII CONTROL K
008C:              3 CTRLL   EQU  140        ;ASCII CONTROL L
008F:              4 CTRLO   EQU  143        ;ASCII CONTROL O
FD1B:              5 KEYIN   EQU  $FD1B      ;MONITOR'S INPUT HANDLER
0038:              6 KSWL    EQU  $38        ;INPUT HOOK ADDRESS
0039:              7 KSWH    EQU  $39
03EA:              8 MVSW    EQU  $3EA       ;ROUTINE TO RECONNECT DOS
00DB:              9 RTBRKT  EQU  219        ;ASCII RIGHT BRACKET
FE89:             10 SETKBD  EQU  $FE89      ;SIMULATES IN#0
00DF:             11 UNDRSCR EQU  223        ;ASCII UNDERSCORE


----- NEXT OBJECT FILE NAME IS NEWKEYS.OBJ0
0300:             13         ORG  $300
0300:4C 0F 03     14         JMP  UNHOOK     ;JUMP TO DISCONNECT ROUTINE
0303:             15 *
0303:             16 * THIS PART ATTACHES OUR ROUTINE INTO THE INPUT HOOK
0303:             17 *
0303:A9 16        18 ATTACH  LDA  #>KEYCHECK ;A= LOW BYTE OF ADDRESS
0305:85 38        19         STA  KSWL
0307:A9 03        20         LDA  #<KEYCHECK ;GET HIGH BYTE
0309:85 39        21         STA  KSWH
030B:20 EA 03     22         JSR  MVSW       ;GO DO IT
030E:60           23         RTS


030F:             25 *
030F:             26 * THIS PART UNHOOKS THE ROUTINE
030F:             27 *
030F:20 89 FE     28 UNHOOK  JSR  SETKBD     ;DO A IN#0
0312:20 EA 03     29         JSR  MVSW
0315:60           30         RTS


0316:             32 *
0316:             33 * THIS IS THE ROUTINE
0316:             34 *
0316:20 1B FD     35 KEYCHECK JSR KEYIN      ;GET THE KEY
0319:C9 8B        36         CMP  #CTRLK     ;CONTROL K?
031B:D0 03        37         BNE  NOTK
031D:A9 DB        38         LDA  #RTBRKT    ;MAKE IT A BRACKET
031F:60           39         RTS
0320:C9 8C        40 NOTK    CMP  #CTRLL     ;CONTROL L?
0322:D0 03        41         BNE  NOTL
0324:A9 DC        42         LDA  #BKSLSH    ;MAKE IT A BACKSLASH
0326:60           43         RTS
0327:C9 8F        44 NOTL    CMP  #CTRLO     ;CONTROL O?
0329:D0 02        45         BNE  CHKDONE
032B:A9 DF        46         LDA  #UNDRSCR
032D:60           47 CHKDONE RTS

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

300:	LDA	#low address of routine		308:	JSR	$3EA	;Reconnect DOS
302:	STA	$38	;Store it in KSWL	30B:	RTS		
304:	LDA	#high address byte of routine		30C:	JSR	$FE89	JSR SETKBD to simulate IN#0
306:	STA	$39	;Store it in KSWH	30F:	JSR	$3EA	;Reconnect DOS
				312:	RTS		

Figure 4

To connect your routine, do a 300G from the monitor. To remove your routine from the hook, do a 30CG.

A Sample Program Using the Input Hook

There are three characters that the APPLE II can understand, but that cannot by typed in from the standard keyboard. They are the backslash (/), the left bracket ([), and the underscore (_). One way to type in these characters is to make a hardware modification to the keyboard. Another way is to attach a routine to the input hook that will convert unused control characters to these characters. This program converts the following characters:

Control K to a left bracket ([)

Control L to a backslash (/)

Control O to an Underscore (_)

To use this program do the following:

Type or BLOAD the program at $300. Note that this program is written for DOS users. If your aren't using DOS, then replace the JMP $3EA with RTS instructions.

To connect the routine, do a 303G from the monitor or a CALL 771 from BASIC.

To disconnect the routine, do a 300G from the monitor or a CALL 768 from BASIC.

The sample program uses the output hook to convert control characters into inverse video characters. All control characters except control M, which is the carriage return, are converted.

```
SOURCE FILE: CONVERT
FDF0:            COUT1   EQU  $FDF0      ;CHARACTER OUTPUT ROUTINE
0037:            CSWH    EQU  $37        ;OUTOUT HOOK HIGH BYTE
0036:            CSWL    EQU  $36        ;OUTPUT HOOK LOW BYTE
008D:            CTRLM   EQU  $8D        ;CONTROL M
003F:            MASK    EQU  $3F        ;MASK TO CONVERT TO INVERSE
03EA:            MVSW    EQU  $3EA       ;RECONNECTS DOS
0080:            NULL    EQU  $80        ;NULL CHARACTER
FE93:            SETVID  EQU  $FE93      ;PERFORMS PR#0
00A0:            SPACE   EQU  $A0        ;SPACE CHARACTER


----- NEXT OBJECT FILE NAME IS CONVERT.OBJ0
0300:                    ORG  $300
0300:4C 0F 03            JMP  UNHOOK
0303:            *
0303:            * ROUTINE TO CONNECT ROUTINE INTO HOOK
0303:            *
0303:A9 16               LDA  #>CONVERT ;GET LOW BYTE OF ADDRESS
0305:85 36               STA  CSWL
0307:A9 03               LDA  #<CONVERT ;GET HIGH BYTE
0309:85 37               STA  CSWH
030B:20 EA 03            JSR  MVSW
030E:60                  RTS


030F:            *
030F:            * THIS UNHOOKS THE ROUTINE
030F:            *
030F:20 93 FE    UNHOOK  JSR  SETVID     ;SIMULATE PR#0
0312:20 EA 03            JSR  MVSW       ;RECONNECT DOS
0315:60                  RTS


0316:            *
0316:            * THIS IS THE CONVERSION ROUTINE
0316:            *
0316:C9 80       CONVERT CMP  #NULL      ;<NULL CHARACTER
0318:90 0A               BCC  GOOUT
031A:C9 A0               CMP  #SPACE     ;>= SPACE CHARACTER
031C:B0 06               BCS  GOOUT
031E:C9 8D               CMP  #CTRLM     ;RETURN CHAR?
0320:F0 02               BEQ  GOOUT
0322:29 3F               AND  #MASK      ;CONVERT TO INVERSE
0324:4C F0 FD    GOOUT   JMP  COUT1

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

Summary of Important Addresses for Using the Hooks

Name	Address	Comment
COUT1	$FDF0	Monitor character output routine.
CSWL	$36	Low address byte of output routine.
CSWH	$37	High address byte of output routine.
KEYIN	$FD1B	Monitor keyboard input routine.
KSWL	$38	Low address byte of input routine.
KSWH	$39	High address byte of input routine.
MVSW	$3EA	Routine to reconnect DOS
SETKBD	$FE89	Simulates a IN#0
SETVID	$FE93	Simulates a PR#0

"COMPUTERS 'R' US"

A CONSUMER COMPUTERS SUBSIDIARY

UNBEATABLE MAIL ORDER DISCOUNTS

apple computer Authorized Dealer

NEW! CALL FOR AVAILABILITY AND PRICES.

$925 FOR **16K**

48K FOR ONLY **$1049**

apple ///

APPLE II OR APPLE II PLUS

APPLE COMPUTER PERIPHERALS

DISK II DRIVE & CONTROLLER CARD With DOS 3.3, List $645	529
DISK II DRIVE & CONTROLLER card	485
DISK II DRIVE ONLY	425
GRAPHICS TABLET	665
SILENTYPE PRINTER w/int. card	515
SSM AIO SERIAL/PARALLEL kit	155
SSM AIO assembled & tested	190
SYMTEC LIGHT PEN SYSTEM	215
SYMTEC SUPER SOUND GENERATOR	225
SVA 8 INCH DISK CONTROLLER CARD	335
VERSA WRITER DIGITIZER SYSTEM	215
VIDEX VIDEOTERM 80 COLUMN CARD	315
VIDEX VIDEOTERM w/graphics ROM	335
LOBO DISK DRIVE ONLY	385
LOBO DRIVE w/controller card	465
DC HAYES MICROMODEM II	319
DAN PAYMAR lower case kit	55

APPLE COMPUTER INTERFACE CARDS

PARALLEL PRINTER Int. card	145
COMMUNICATION CARD w/conn. cable	185
HI-SPEED SERIAL Int. card	145
LANGUAGE SYSTEM with PASCAL	425
CENTRONICS PRINTER Int. card	185
APPLESOFT II FIRMWARE card	149
INTEGER BASIC FIRMWARE card	149

MOUNTAIN HARDWARE ACCESSORIES
A Division Of Mountain Computer

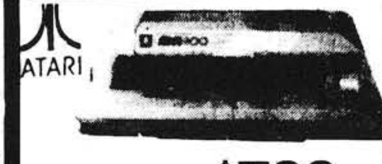
APPLE CLOCK/CALENDAR card	225
SUPERTALKER SD200 SPEECH SYNTHISIZER SYSTEM	245
ROMPLUS w/keyboard filter	185
INTROL/X-10 BSR REMOTE CONTROL SYSTEM	245
INTROL/X-10 controller card only	165
ROMWRITER SYSTEM	155
MUSIC SYSTEM(16 voices/stereo)	465
A/D-D/A 16 CHANNELS	319
EXPANSION CHASSIS (8 slots)	555

APPLE ADD-ONS

CORVUS 10 MEGABYTE HARD DISK DRIVE SYSTEM w/pwr supply	4395
CORVUS CONSTELLATION	595
16K MEMORY UPGRADE KIT (TRS-80, APPLE II, SORCERER)	60
ABT NUMERIC INPUT KEYPAD (specify old or new kybrd)	115
ALF MUSIC SYNTHISIZER	235
BRIGHTPEN LIGHTPEN	32
GPIB IEEE-488 (1978) Int.	259
ARITHMETIC PROCESSOR card	335
SPEECHLINK 2000 (64 Word Vocab.)	215
M&R SUP-R—MOD TV MODULATOR	30
MICROSOFT Z-80 SOFTCARD SYSTEM w/CP/M & MICROSOFT BASIC	299
MICROWORKS DS-65 DIGISECTOR	339
LAZER lower case adapter	50
M&R SUPER TERMINAL 80 column card	335

APPLE II or APPLE II PLUS SOFTWARE

PASCAL with LANGUAGE SYSTEM	425
FORTRAN for use with LANGUAGE SYSTEM	165
CP/M for use with MICROSOFT Z-80 SOFTCARD (Incl.)	299
DOS 3.3	49
THE CONTROLLER General Business System	519
THE CASHIER Retail Management & Inventory System	199
APPLEWRITER Word Processor	65
APPLEPOST MAILING list system	45
APPLEPLOT Graph & Plot Sytem	60
DOW JONES PORTFOLIO EVALUATOR	45
APPLE CONTRIBUTED VOLUMES 1 thru 5 w/manuals	30
VISI-CALC by PERSONAL SOFTWARE	120
DESKTOP/PLAN by DESKTOP COMPUTERS	85
CCA DATA MANAGEMENT SYSTEM By PERSONAL SOFTWARE	85
APPLEBUG ASSEMBLER/DISASSEMBLER	75
APPLE DOS TOOL KIT	65



16K FOR **$799**

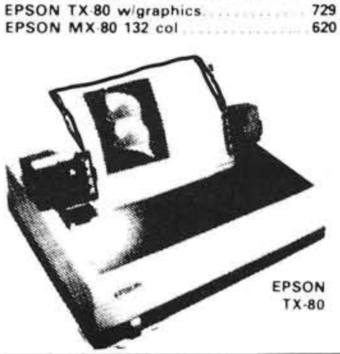
ATARI 800
PERSONAL COMPUTER SYSTEM

ATARI ACCESSORIES

400 COMPUTER	479
820 PRINTER (40 col)	459
810 DISK DRIVE	559
410 Program Recorder	59
815 DUAL DISK DRIVE	1199
822 THERMAL PRINTER (40 col)	369
825 PRINTER (80 col. imp.)	795
850 INTERFACE MODULE	175
ATARI 16K RAM MODULE	155
LIGHT PEN	65
ACOUSTIC MODEM (CAT)	169
COMPUTER CHESS	35
SPACE INVADERS	19
STAR RAIDERS	49
SUPER BREAKOUT	35
3-D TIC-TAC-TOE	35
VIDEO EASEL	35
MUSIC COMPOSER	49

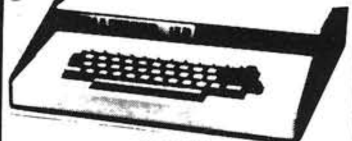
PRINTERS

ANADEX DP-8000	775
ANADEX DP-9500	1350
BASE 2	599
CENTRONICS 737	825
MPI 88-T	699
PAPER TIGER IDS-440 w/graphics	895
NEC SPINWRITER	2550
TRENDCOM 200	519
SILENTYPE w/Int.	515
EPSON TX-80 w/graphics	729
EPSON MX-80 132 col	620



EPSON TX-80

OHIO SCIENTIFIC



$699

C4P
8K ROM BASIC
8K RAM EXPANDABLE TO 96K
32x64 UPPER & LOWER CASE
256x512 GRAPHICS POINTS
PROGRAMMABLE TONES
ANALOG INPUTS
C4PMF (1 DISK DRIVE) 1599

NEW!

$429

CIP MOD II
8K ROM BASIC
8K RAM EXPANDABLE TO 32K
COLOR EXPANSION
48 LINE DISPLAY EXPANSION

SOFTWARE

	Cassette	Disk
SPACE INVADERS	19	29
SARGON II	30	35
FORTH	N/A	69
OS-65-D V3.3	N/A	79
MDMS PLANNER	N/A	100
GRAPHICS I	N/A	35
DAC I	N/A	45
ASSEMBLER/EDITOR	40	N/A
EXTENDED MONITOR	20	N/A
PASCAL & FORTRAN (4P & 8P only)		450

When ordering please specify system.

PLOTTERS

only **$1095**

WATANABE MIPLOT

for more info please call or write

VIDEO MONITORS

LEEDEX VIDEO 100	129
SANYO 9" B&W	165
SANYO 15" B&W	245
PANACOLOR 10" COLOR	329
NEC 12" HI-RES COLOR	875
NEC 12" LO-RES COLOR	399
NEC 12" GREEN PHOSPHER(P31)	239

$129

LEEDEX VIDEO 100

master charge | VISA | AMERICAN EXPRESS

- FAST DELIVERY
- LOW PRICES
- COURTEOUS SERVICE
- KNOWLEDGEABLE STAFF
- LARGE VARIETY

IN CALIFORNIA, OR FOR BACKORDER OR TECHNICAL INFO CALL:
(714) 698-8088

TOLL FREE ORDER LINE: 1-800-854-6654

CREDIT CARD USERS PLEASE READ TERMS OF SALE IN ORDERING INFORMATION

ORDERING INFORMATION: Phone Orders invited using VISA, MASTERCARD, AMERICAN EXPRESS, or bank wire transfers. VISA & MC credit card service charge of 2%, AE credit card service charge of 5%. Mail orders may send charge card number (include expiration date), cashier's check, money order or personal check (allow 10 business days to clear.) Please include a telephone number with all orders. Foreign orders (excluding Military PO's) add 10% for shipping and all funds must be in US dollars. Shipping, handling and Insurance in U.S. add 3%. California residents add 6% sales tax. Our low margins prohibit us to send COD or on account. All equipment subject to price change and availability. Equipment is new and complete with manufacturer warranty. We ship most orders within 2 days. Order desk hours are Monday thru Saturday 9-5 PST. Send for FREE 1981 Catalog.

WE ARE A MEMBER OF THE BETTER BUSINESS BUREAU AND THE CHAMBER OF COMMERCE — RETAIL STORE PRICES MAY DIFFER FROM MAIL ORDER PRICES.

PLEASE SEND ORDERS TO: CONSUMER COMPUTERS MAIL ORDER CRU Division 8314 PARKWAY DRIVE, GROSSMONT SHOPPING CENTER NORTH, LA MESA, CALIFORNIA, 92041

An Ultra-Fast Tape Storage System

A simple hardware modification to the Ohio Scientific Superboard and the use of a good home hi-fi tape recorder yield data-transfer rates of up to 9600 baud.

John E. Hart
5 Marvin Road
Wellesley, MA 02181

Why Tape?

Most hobbyist micros come with a simple, but slow, bulk storage system using a dictaphone-type cassette tape recorder. Because of the rather low reproduction quality of such machines, data rates typically are 300 to sometimes 600 baud (bits per second). For transferring short programs or data files between tape and memory, this is often sufficient. However, at 300 baud, for example, it takes about 1.5 seconds to load a typical line of a BASIC source code. If the program contains only a few lines, the times involved are not objectionable.

Recently, I was working on a compiler for my Ohio Scientific Superboard. It inputs BASIC statements and writes object machine code in the high end of memory. Needless to say, this program, itself written in BASIC, was long. It took almost 15 minutes to load its 350 lines from cassette. Many computer owners faced with similar problems might go to the obvious means of enhancing program retrieval—the disc.

Unfortunately, for many purposes, there are severe limitations with existing disc systems. These limitations, coupled with the fact that I already owned a good hi-fi tape deck, led me to develop a simple, high-speed tape storage system that transfers 16K of BASIC code from tape to RAM in about 15 seconds! This system is almost competitive with disc systems and has several advantages. The hardware and software required are so simple, it would be easy for anyone owning a good tape recorder to adopt the high-speed system. Since a disc drive costs as much or more than a tape recorder, some people might opt for the latter, and buy a piece of equipment with multiple uses.

Disc systems are of limited use in jobs that require either a large amount of RAM or that require fast execution. For fast execution, I purchased the Ohio Scientific Superboard, because I know of a simple jumper connection that doubles the speed (see the article by J.R. Swindell, "The Great Superboard Speed-Up," MICRO, February 1980, 21:31). Increasing the clock from 1 to 2 Mhz was very important to me, since I do a lot of lengthy calculations. Unfortunately, Ohio Scientific disc systems will not *run* at 2 Mhz without major hardware surgery, and software modification as well. Worse still, the Ohio Scientific disc-operating system 65V uses 9-digits precision arithmetic. This is really not any more useful to me than the standard 6½-digit precision, and moreover, it runs about 50% slower. So in summary, using a disc would cause my jobs to run almost three times more slowly than with tape and normal BASIC in ROM.

I do a lot of calculations on large two-dimentional arrays. Thus, in addition to speed, I need a large amount of RAM for immediate storage. Since typical disc-operating sysems occupy 12K or more of RAM, the execution time is further slowed by the necessity for repeated transfer of 10K blocks of data between RAM and disc. In total, it seemed as if any gain in program and data transfer using the disc would be offset by slow execution. Wouldn't it be nice to store my programs an/or object codes on tape and to transfer them into memory at a rate approaching the upper end of the frequency response of the tape drive? Since a good hi-fi cassette deck with Dolby reaches 10kc in its response, and a good reel-to-reel deck goes above 20kc, theoretically, it ought to be possible to squeeze 4800 to 9600 baud out of these units.

What is Kansas City Standard Format?

Most computers come with a tape system called Kansas City Standard. In this format, ones and zeros are represented on tape by two different frequencies. This is done because frequency modulation is much less sensitive to noise and tape alignment errors than amplitude modulation, where zeros might be represented by a zero signal and ones by a single pulse or frequency. In fact, a zero is recorded as 8 cycles of a frequency 8 times the baud rate, and ones are represented by 4 cycles of a frequency 4 times the baud rate. Thus, at 300 baud, zeros are short bursts of 2400 hz signal, and ones are short bursts of 1200 hz oscillations.

Baud Rate	Byte Rate (approx)*	0-frequency	1-frequency
300	30/sec	2400 hz.	1200 hz.
600	60	4800	2400
1200	120	9600	4800
2400	240	18200	9600
4800	480	36400	18200
9600	960	72800	36400

*This depends on the word structure: 7-bit, no parity; 8-bit, with parity, etc.

Table 1

It is perhaps obvious that, if this technique is reliable (and it certainly works very well with cheap recorders), you could try to increase the baud rate simply by employing a tape recorder with a better frequency response. This is in fact the case. A dictaphone-type machine can reliably handle 600 baud (PET already does this); a hi-fi cassette can do 1200 baud; and a good reel-to-reel, operating at 7½ or 15 ips, can do 2400. However, as table 1 indicates, getting much faster data transfer than 2400 baud, with even the best reel-to-reel tape recorder, is probably impossible within the framework of the Kansas City format. The required frequency response is just too high. No audio tape machine has much usable response above 25 kc.

Although it turns out that some gains could be made in Kansas City format by using a good tape machine, unfortunately, in loading BASIC programs from tape, or in loading machine code using the Ohio Scientific there are stumbling blocks. The Microsoft BASIC interpreter does a considerable amount of data massaging as each line of BASIC is loaded. This takes time. A lot of time! Input lines are decoded, and certain errors are trapped and can appear on the screen while a program is being loaded, before it has RUN. The 1-Mhz Superboard will load 600-baud tapes *if* they are recoded with 8 nulls (for example, NULL8, SAVE, LIST), but falters at 1200 baud. However, if the clock is flipped up to 2 Mhz, the 1200-baud tapes load well, but the 2400-baud tapes fail. Thus fast tape loading cannot be done with Microsoft BASIC. However, it can be done using a simple machine code loader and saver described below. But first, I must outline a simple trick that gets up to 9600 baud with a 20kc response deck.

How Is Kansas City Standard Data Decoded?

Recall that in Kansas City format a zero is 8 cycles and a one is 4. Then, to eliminate or minimize noise, one might simply count the pulse train. A count of more than 6 pulses per bit width (1/baud rate) would be a zero, a count of less than seven could be one. (You may actually use a counter or, as APPLE does, use a phase-locked loop.) Thus, an extra or dropped cycle would not have much effect. However, this is not how Ohio Scientific decodes. In figure 1, *a* shows a typical input pulse train obtained by taking the tape play signal and amplifying it beyond the clipping point. In the Superboard, this pulse train is fed into a retriggerable one-shot multivibrator. This device triggers (output goes high) on the positive going edge of each input pulse. The output then stays high for a time dependent on an RC circuit (R_1 and C of figure 2). Since it is retriggerable, if another input pulse arrives while the output is high, a new time-delay cycle is started. The time delay ($t_d = R_1 \times C$) is chosen so that retriggering occurs for the higher frequency input (zero), but not for the lower frequency (one). This is shown in *b* of figure 1. You can see that a certain amount of noise immunity is afforded here, in that tape jitter or pulse stretching has to occur for a fairly long time (1/0-frequency) before the trigger errors occur. The one-shot output is sampled by the serial communications adaptor at the end of the bit input as shown in *c* of figure 1.

Actually, there is a little more circuitry in-between, but it is not important for our purposes. Most of the data is irrelevant to the final decoding. It is only the last set of pulses just before the sample that determines whether a one or a zero is recognized! This would not be the case if the counting scheme I suggested above had been used. But this shows that by substituting real data in place of the first ¾ of unused pulses, you could multiply the data density and transfer rate by a factor of four. Thus, we could go from 2400 to 9600 baud, while still operating at a maximum frequency of 18.2 kc. You might think that doing this would be just asking for read errors. In truth, for a given program length and 0-frequency, the error probability is unchanged.

How to Change Baud Rates and Quadruple Data Density

It is necessary to install a 3-pole,

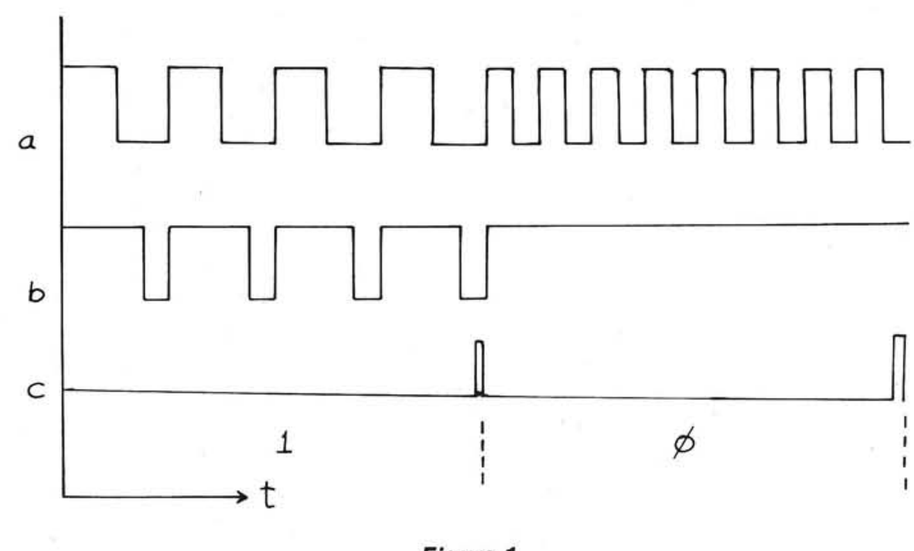


Figure 1

5-position switch, connected as shown in figure 2. This assumes that the reader will want all the options:

1. Normal Kansas City recording at 300 baud (position a)

2. Normal Kansas City recording at 600 baud (position b)

3. Normal Kansas City recording at 1200 baud (position c)

4. Bi-mod* recording at 4800 baud (position d)

5. Bi-mod recording at 9600 baud (position e)

*This is what I call the scheme where a zero is 2 cycles (instead of 8), and a one is 1 cycle (instead of 4), or twice the period.

The first three positions give a straightforward modification to the 8- and 4-cycle Kansas City Standard record/play technique. I also retain these modes in my machine, so I can load cassettes recorded this way into my computer and make fast tapes for rapid loading. Also, although I rarely get a read error at 9600 baud, I like to feel secure, knowing I have a backup cassette—just in case... The circuit also includes switch positions for both 4800 and 9600 baud. A good Dolby cassette deck is capable of 4800 but not 9600. I have tried three reel-to-reel decks at 7½ ips and they all worked at 9600, but I cannot guarantee that all units of varying condition will. Thus, if a reader doesn't want to wire in all these options, I would suggest that at least option 1 or 2 and option 4 and/or 5 be included.

Referring to figure 2, the first pole of the switch (S_a) just taps off the main Superboard clock divider U59 and U30, to send different clock pulses to the serial data transmitter/receiver (ACIA). The second pole (S_b) selects the apropriate time delay for the retriggerable one-shot U69, corresponding to the clock frequency selected by S_a. R_{1a} and R_{1b} as well as C come with the Superboard and are set for 300 baud. Fixed resistors may be used for 600 and 1200, since the device is not very sensitive. I put trimpots in for the higher baud rates, and you might want to do this for all the positions and then set them by trial and error (for example, load a program) to the middle of the acceptance band.

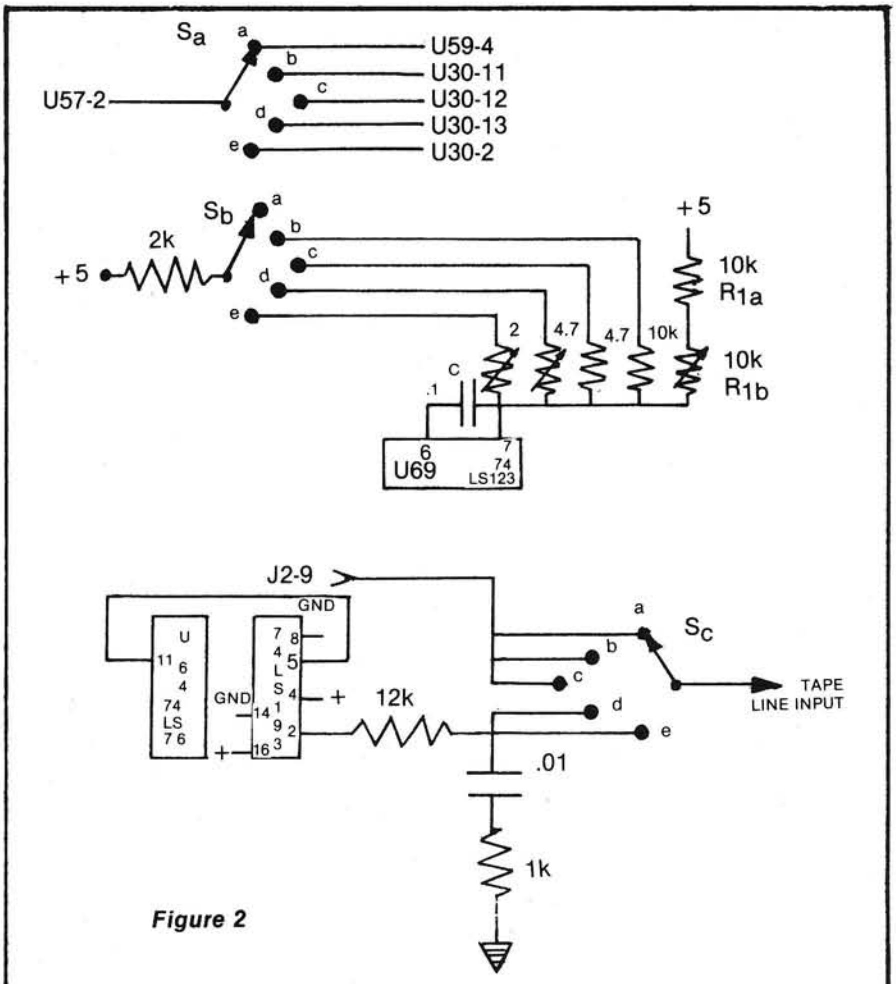


Figure 2

To get the bi-mode of recording, we take the normal 8- and 4-cycle modulation coming out of U64-11, and count it down by a factor of 4 using a 74LS193 counter. Thus, to get 9600 baud (from 300), I have increased the clock to the ACIA by a factor of 32, but divided the 8- and 4-cycle outputs down to 2 and 1, so that the frequency of the signal going on the tape only increases by a factor of 8. I found it easiest to mount the counter in one of the unused prototype sockets on the Superboard. Finally, pole S_c of the switch selects between normal 8/4-cycle modulation and 2/1, as shown, and feeds the transmitted data to the line input of the tape deck.

Software

To get started at 4800 or 9600 baud, I included a pair of simple machine code programs to store and read tapes. Dump all of the RAM below a given page number and down to ADDR 0000 onto tape. Then, to load a BASIC program, read the tape back in, and because it includes pages zero and one with all the BASIC tables and flags exactly as they were just before recording, the program comes in all ready to run.

You can build on this software. For example, every time you hit BRK-W or BRK-C, a command is sent to the ACIA to format output as 8 bits of data followed by 2 stop bits. With this high-speed scheme, it might be good to command the ACIA to output 8 bits of data, one parity, one stop bit, and check parity on reload. However, you can tell if there is a read error (usually), since for BASIC program, the loader should end up exactly at location 0000 with a 4C hex there. Unless this is the case, a bit has been dropped.

In fact, I have only been able to cause this to happen by making the tape play volume much too small, or by rather heavily touching the reels as the tape is playing back! You may also want to relocate these routines to the back of your memory.

Some Hints

This system obviously approaches the limits of standard audio recorders and tape. However, I emphasize again that I have yet to misload a long program that was properly recorded at 9600 baud. The usual precautions should always be taken. Maintain *clean* tapes and heads. Demagnetize (the heads!). Use the best back-coated, extended-range tape you can get (I like Maxell UD-XL), and do not rerecord over old material. Put each new program on brand new tape to avoid print through. At 7½ ips, the data density is so high that you get about 20 megabytes out of a 1200-ft. tape, so there is little sense in not doing this. Make sure the equalization is correct, and set the record level for optimum high frequency response (an oscilloscope is useful here).

I hope this technique opens up new horizons. With a stereo recorder, you may immediately think about using the other channel for search or file headers. In any case, the ability to load 16K programs in 15 to 30 seconds, and to have all of RAM available for user storage and programs is an enormous advantage in many situations. And once you have loaded your program, you can switch over, relax, and enjoy some hi-fi!

To save pages 17 to zero run (G):

```
                    HIGH SPEED WRITE

1936                          ORG    $1936

1936 A9 FF                    LDAIM  $FF     SET UPPER MEMORY START FOR DUMP
1938 8D 48 19                 STA    $1948   LOAD INTO ADDRESS POINTER LOCATION
193B A9 17                    LDAIM  $17     SET UPPER MEMORY PAGE START
193D 8D 49 19                 STA    $1949   LOAD INTO ADDRESS POINTER LOCATION
1940 AD 00 F0                 LDA    $F000   LOAD ACIA STATUS REGISTER
1943 29 02                    ANDIM  $02     MASK BUSY BIT
1945 F0 F9                    BEQ    $1940
1947 AD FF FF                 LDA    $FFFF   FILLED IN BY CODE ABOVE
194A 8D 01 F0                 STA    $F001   WRITE TO ACIA
194D CE 48 19                 DEC    $1948   DECREMENT LOW BYTE OF ADDR.
1950 D0 EE                    BNE    $1940
1952 CE 49 19                 DEC    $1949   DECREMENT HIGH BYTE
1955 10 E9                    BPL    $1940
1957 4C 00 FE                 JMP    $FE00   JUMP TO MONITOR
```

To read in or load pages 17 to zero run:

```
                    HIGH SPEED READ

18F0                          ORG    $18F0

18F0 A9 FF                    LDAIM  $FF
18F2 8D 11 19                 STA    $1911
18F5 A9 17                    LDAIM  $17
18F7 8D 12 19                 STA    $1912
18FA AD 01 F0                 LDA    $F001   READ ACIA TO CLEAR IT
18FD AD 01 F0                 LDA    $F001
1900 AD 00 F0                 LDA    $F000   RECEIVE STATUS CHECK
1903 29 01                    ANDIM  $01
1905 F0 F9                    BEQ    $1900
1907 AD 01 F0                 LDA    $F001   READ ACIA
190A 8D 10 D2                 STA    $D210   WRITE CHAR RECEIVED TO SCREEN
190D 8D 05 D2                 STA    $D205
1910 8D FF 17                 STA    $17FF   WRITE DATA TO MEMORY
1913 CE 11 19                 DEC    $1911   DECREMENT LOW BYTE OF ADDR.
1916 D0 E8                    BNE    $1900
1918 CE 12 19                 DEC    $1912   DECREMENT HIGH BYTE
191B 10 E3                    BPL    $1900
191D 4C 00 FE                 JMP    $FE00   JUMP TO MONITOR
```

µ

Skyles Electric Works

BASIC Programmer's, Toolkit™, Disk-O-Pro™, Command-O™

For PET™ Owners Who Want More Fun And Fewer Errors with Their Programs

Here are thirty-five commands you'll need, all on dual chips you can install in two minutes without tools, **on any PET or PET system.** 2KB or 4KB of ROM firmware on each chip with a collection of machine language programs available from the time you turn on your PET to the time you shut it off. No tape to load or to interfere with any running programs.

For PET/CBM 2001-8, -8N, -16N/B, -32N/B, 3016 and 3032

BASIC Programmers Toolkit™ commands

AUTO^{ed} DELETE^{ed} RENUMBER^{ed} HELP^{ed} TRACE^{ed} STEP^{ed} OFF^{ed} APPEND^{ed} DUMP^{ed} FIND^{ed}

BASIC Programmers Disk-O-Pro™

CONCAT^{B80} DOPEN^{B80} DCLOSE^{B80} RECORD^{B80} HEADER^{B80} COLLECT^{B80} BACKUP^{B80} COPY^{B80} APPEND^{B80} DSAVE^{B80} DLOAD^{B80} CATALOG^{B80} rename^{B80} SCRATCH^{B80} DIRECTORY^{B80} INITIALIZE^{BS} MERGE^{BS} EXECUTE^{BS} SCROLL^{ed} OUT^{ed} SET^{ed} KILL^{ed} EAT^{ed} PRINT USING^{BS} SEND^{BS} BEEP^{BS}

For CBM 8016 and 8032

BASIC Programmer Command-O™

AUTO^{ed} DUMP^{ed} DELETE^{ed} FIND^{ed} (improved) **HELP^{ed} KILL^{ed} OFF^{ed} TRACE^{ed}** (improved) **RENUMBER^{ed}** (improved) **INITIALIZE^{BS} MERGE^{BS} MOVE^{BS} EXECUTE^{BS} SCROLL^{ed} OUT^{ed} SET^{ed} SEND^{BS} PRINT USING^{BS} BEEP^{BS}**

ed — a program editing and debugging command
B80 — a BASIC command also available on Commodore CBM™ 8016 and 8032 computers.
BS — a Skyles Electric Works added value BASIC command.
BASIC Programmers Toolkit™ is a trademark of Palo Alto IC's.
BASIC Programmers Disk-O-Pro™, Command-O™ is a trademark of Skyles Electric Works.
PET™, CBM™ are trademarks of Commodore Business Machines.

Can be placed in main board socket or with precision-engineered interface

Phone or write for information. We'll be delighted to answer any questions and to send you the complete information package.

Skyles Electric Works

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

SYM-Bell

Have you ever looked fondly at a telephone memory dialer at your local electronics supply store, but decided that the $100 plus could be put to better use elsewhere? Well, for about $10 your SYM-1 or similar computer can do all that commercial dialers do, plus much more.

Randy Sebra
54 Krouse Court
Aberdeen, MD 21001

Commercially available,dedicated telephone dialers can cost up to one-third the price of the single board SYM-1 equipped with BASIC. However, with the addition of a simple relay interface, costing less than $10 and driving software, the SYM-1 can out perform any of these units. The combination of machine language for control, and BASIC for flexibility yields an extremely powerful system. Unlike the commercial systems which are usually limited to a maximum of 32 numbers, the numbers available to the SYM-1 are a function only of the available user memory. Also, this system is capable of doing things beyond the scope of most commercial dialers.

There have been a number of articles in periodicals and books on telephone dialing by computer. However, these have been describing dialing by using the microprocessor to generate Touch Tone™ digits. The only problem with this method is that the telephone system accessed must be compatable with the Touch Tone™ dialing system, and not all areas of the country (or even all areas within any one locality) have this capability. The interface described here generates dial pulses, which are compatable with any system.

Additionally, this method is not restricted to the SYM-1. Virtually any microcomputer with a single available output port that can be accessed by BASIC can be adapted to do this operation.

Dialing is Simple

Most home telephones use a three wire system. The two line leads are usually the red (ring) and green (tip) wires. The line leads carry the analog conversation signal as well as dialing information for either Touch Tone™ or rotary dialing and the ring signal for the bell. A third wire, not shown (usually the yellow lead) serves as a ground reference. The fourth, fifth, and sixth leads on the current modular plug are not normally used for home systems.

When not in use, the telephone receiver is disconnected from the line by the normally open cradle switch. When the handset is picked up, the cradle switch closes and connects the receiver to the telephone line. The remote switching station senses the active current loop (20 to 40mA at 24 VDC), and issues the familar dial tone. When the rotary dial is used, it repeatedly breaks and makes a connection in one of the line leads. It is this break/make action that creates dial pulses sensed by the switching station which routes the call to the proper destination.

You can do a simple experiment to see how this works. Pick up the handset from your telephone. Then rapidly tap one of the cradle switch buttons four times, pause, tap once, pause again, and rapidly tap once more. If done properly, you should get a ring, and directory assistance will answer.

Interfacing With the Telephone

Therefore, by putting a normally closed relay in one of the line leads, dial pulses can be generated by toggling the relay properly. This is shown in figure 1. The circuit takes advantage of the fact that on the SYM-1, the PB4-PB7 outputs from the #3 6522 VIA are buffered so that each is capable of driving a 5-volt relay. The buffer for PB7 is shown in the dotted section of figure 1. The diode across the relay is for transient protection. The voltage variable resistor prevents high voltage transients from being introduced into the phone line, but more importantly, prevents a high voltage transient from a thunder storm or other sources from getting into the interface from the phone lines and wiping out the computer. An alternative to the resistor is two Zener diodes back to back across the lines. The ring signal for the telephone bell is approximately 90VAC, 20HZ at .5 Amps, so the relay used should be able to withstand this in case an incoming call comes in inadvertantly while the interface is active.

At this point, it should be mentioned that although the relay and voltage variable resistor isolate the circuitry from the phone line, this direct connection should only be used for privately owned systems. Such a connection to TELCO lines is "illegal", and both the FCC and TELCO frown of this type of connection. There is, however, a way to make a "legal" connection of this type of interface. This will be

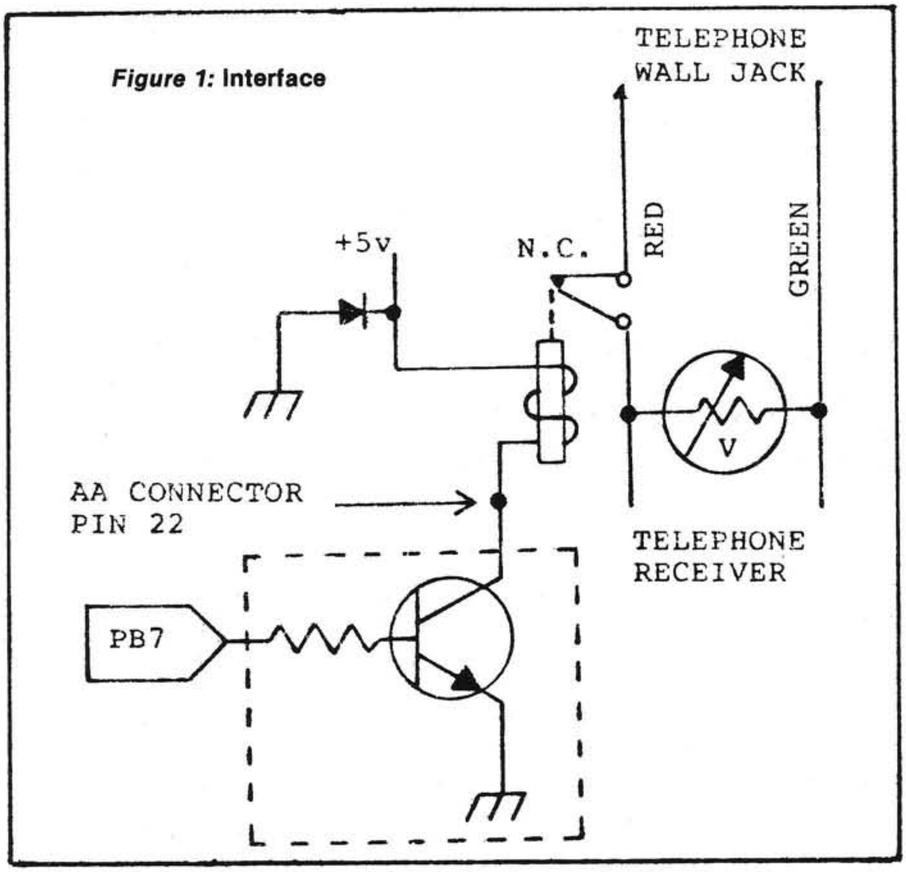


Figure 1: **Interface**

discussed later, after the interface and software operation are fully understood.

Machine Language Routines

Table 1 presents the timing requirements for dial pulses. The pulses are sent out in groups depending on the digit dialed; that is, a single pulse for a "1", two pulses for a "2" and so on with ten pulses for an "0". There must be a pause between each digit dialed as shown in the table. In some areas dialing can be done at a 20 Hz rate, and all that is needed is a change in the software timing. Notice that there is a wide tolerance range in the timing requirements. After all, the conventional spring-driven rotary dial that works so well is not quite a crystal controlled pulse generator. For this reason, although the timing loops in the software are close to the nominal values, extreme attention to detail such as counting delays incurred by JSR's and other instructions in machine language and execution times in BASIC was not taken.

Listing 1 presents the machine language routines which drive the relay and operate the elapsed timer. Notice that the general delay routine DELAY uses timer 2 of one of 6522 VIA's in the one shot mode. The routine allows a continuously variable time delay from a few microseconds to over two minutes. The variability is needed for the generation of a number of different delays, including the long .8 sec delay used in the hang up routine, HANG.

The second timing routine which uses the 6522, TIMER, is an interrupt driven routine which uses timer 1 in its free running mode, but with the PB7 output disabled by the setting of the ACR. The routine uses three page zero addresses, $F0, $F1 and $F2, which do not conflict with the operation of the BASIC program. The interrupt routine it refers to, UPDATE, is similar to many published real time clock routines. It is shorter, however, because there is no need to keep track of hours and it will count up to 99 minutes and 59 seconds before resetting to zero. For telephone conversations, this should be more than sufficient—except perhaps if you have a teenage daughter at home as I do.

When called from BASIC, the routine initializes the timer and sets up the output vector for the onboard LED display. The interrupt routine updates the count of minutes and seconds and outputs the elapsed time each second. The main routine constantly scans the LED display until it senses that a key on the terminal has been pressed. At that time, it stops the timer, resets the output vector, does a hang up and returns control back to BASIC.

The initialization routine is needed because the PB4-PB7 lines are configured on power up and reset as input ports, which would turn the relay on and de-activate the telephone line. The interface should be disconnected from the telephone lines at all times except when the SYM is being used as a dialer, since other SYM programs might use PB7 as an input/output port.

The hang up routine is just a disconnect for the same duration as an interdigital pause. On some systems, this may not be long enough to effect a disconnect, and the time may have to be set as long as two or three seconds.

The dialing routine is merely a variable count pulse generator which generates the proper number

	Nominal value	Range
Pulse rate	10pps	8-11pps
Break time	61%	58-64%
Interdigit pause	800msec	600msec -3 sec

Table 1: **Timing Requirements**

of pulses for the digit requested with the proper timing and duty cycle.

The machine language routines were written to occupy the high memory for a 4K system. They can be easily relocated, however, by changing the values which are underlined in the listing.

The BASIC Program

The interface and machine language routine, as presented so far are of rather limited value. Here is where the versatility of a BASIC driving program comes into play. Listing 2 presents one such program. With this program, it is possible to not only dial a single number with redial capability, but to sequentially dial any combination of numbers from the directory in virtually any order, all with redial and selective hang up capability. Additionally, any call can be timed, with the elapse time being continuously displayed on the on-board LED display, and the total elapsed time printed out at the terminal.

Additionally, the numbers can contain an access pause, identifies by a "." in the number. An access pause is needed when the dialing of one telephone number results in a dial tone for a second number. The most familiar example of this is item 18 in the directory—getting an outside line from a business phone. The digit "9" is dialed, and when the dial tone is obtained, the number is dialed. The program does this operation automatically. The "9" is dialed and the program waits for an entry to dial the rest of the number. For a busy signal, the redial can come after either number. A second example is that of a call diverter used in directory item 19. Some large time-shared computers use this type of set up. A local number is dialed, and a call diverter routes the call to another exchange within the local area of the computer. A second number is then called for the final connection. This type of operation is often much cheaper than a single toll call.

This program, including the machine language routines, in a minimal 4K system can store up to 50 numbers-depending on the length of the numbers and the

***Table 2:* Sample Run**

```
.J 0
MEMORY SIZE? 3866
WIDTH?

 3353 BYTES FREE

BASIC V1.1
COPYRIGHT 1978 SYNERTEK SYSTEM CORP.

OK
SAVE X
SAVED

OK
LOAD M
LOADED

OK
LOAD D
LOADED

OK
RUN
 0    POLICE         1    FIRE
 2    DOCTOR         3    LAWYER
 4    SCHOOL         5    PARENTS
 6    WORK           7    WIFE'S WORK
 8    NEIGHBOR       9    BROTHER
 10   JANE           11   JOE
 12   JOHN           13   SALLY
 14   JIM            15   JOAN
 16   DORIS          17   BILL
 18   HOME           19   COMPUTER

FIRST PICK UP RECEIVER AND WAIT FOR DIAL TONE.
ENTER THE DIRECTORY NUMBER(S) YOU WISH TO DIAL. YOU MAY
ENTER A SINGLE NUMBER, A SEQUENCE OF NON-CONSECUTIVE
NUMBERS SEPARATED BY SEMI-COLONS, OR A RANGE OF
NUMBERS SEPARATED BY A DASH.

ANY TIME YOU WISH TO HANG UP, ENTER AN H. TO RE-DIAL THE
PREVIOUS NUMBER, ENTER AN R(HANG UP NOT NECESSARY). TO
CONTINUE AFTER AN ACCESS PAUSE, ENTER C(OR H OR R IF THE
LINE IS BUSY). TO USE TIMER, ENTER A T AFTER THE
CALL IS ANSWERED. WHEN THE CONVERSATION IS OVER,
PRESS ANY KEY TO STOP TIMER AND HANG UP.

READY  3
DIALING  LAWYER 555-3958? R
DIALING  LAWYER 555-3958? R
DIALING  LAWYER 555-3958? H
RUN AGAIN(Y OR N) Y
READY  5;9
DIALING  PARENTS 1-804-559-6741?  T
Y
 ELAPSED TIME : 4 MINUTES AND 15 SECONDS
DIALING  BROTHER 1-703-556-0924?  T
P
 ELAPSED TIME : 2 MINUTES AND 58 SECONDS
RUN AGAIN(Y OR N) Y
READY  10-17
DIALING  JANE 555-0226? H
DIALING  JOE 555-9328? H
DIALING  JOHN 555-1293? H
DIALING  SALLY 555-3092? H
DIALING  JIM 555-8876? H
DIALING  JOAN 555-2783? H
DIALING  DORIS 555-5638? H
DIALING  BILL 555-9951? H
RUN AGAIN(Y OR N) Y
READY  19
DIALING  COMPUTER 555-4900ACCESS PAUSE - USE C,H OR R OPTIONS.C
554-1200? T
G
 ELAPSED TIME : 20 MINUTES AND 33 SECONDS
RUN AGAIN(Y OR N) N

OK
```

length of their identification in the directory. This number can be increased by removing the REMARK statements and using multiple statement lines. Further, if the memory is expanded to 8K, another 200-300 numbers can be added.

Although the program is commented, it's use is best demonstrated by an example. Table 2 shows a sample run. The memory size of 3866 is for a 4K system. The dummy call, SAVE X at the start of the program (with the tape recorder off) is necesary to overcome the fact that when first entering BASIC the system RAM is still write protected, preventing a tape load to operate properly. The machine language routines were saved as file number $4D, allowing it to be read in by BASIC as file "M". This is always done *before* loading the BASIC saved dialer program file "D", since a LOAD command causes BASIC to do a NEW, which wipes out any current BASIC program.

When the program is started by the RUN statement, it prints the full directory list and complete instructions. After one complete set of operations, the program cycles as long as desired without reprinting the directory or instructions. More numbers can be added to the program if the code for these printings is eliminated.

The first run is that of dialing a single number, "LAWYER". On the first two tries at dialing, the number is busy, and an "R" is entered for a redial. On the third try, the call goes through, and when the conversation is over, an "H" is entered to terminate the connection. The program then cycles for another run.

The second run illustrates the use of the timer and dialing two nonconsecutive numbers, "PARENTS" and "BROTHER". Since both numbers are long distance calls, a "T" is entered after each call goes through. This starts the elapsed timer, and the elapsed time is continuously displayed on the LED display. When the call is over, any key is pressed (This is illustrated by using a "Y" the first time and then a "P".) Then the program does a hang up and dials the next number.

The third run demonstrates dialing consecutive numbers. In this example, items 10 through 17 ("JANE" through "BILL") are called, one after another, until some action has been taken on all of the numbers. Any number of redials ("R" option) could have been done during this sequence.

The last example shows the dialing of a single number containing an acess pause, " COMPUTER ". The timer is also used to keep track of connect time to the computer. Again, any number of redials could be used anywhere until the connection is completed.

The "Legal" Connection

There are two approaches to connecting the interface to the telephone without fear of a hassle from the telephone company. The first, although inelegant, is quite effective. If the voltage-variable resistor is removed and the relay is replaced by a solenoid, numbers may be dialed by pulsing the cradle switch directly as in the experiment earlier in the article. There are several problems with this approach, however. First, the solenoid has to be mounted on the telephone with a rather close tolerance so that at one end of its travel the cradle switch is fully closed and at the other end of its travel, the cradle switch is fully open. This limits its use by having to be mounted permanently on the telephone, thereby limiting the telephone's use or else a very accurate, repeatable mounting device must be constructed and set up every time the dialer interface is to be used. Lastly, since the cradle switch is designed only to be used to initiate, answer or terminate a call, it may take quite a beating by repeatedly dialing numbers in this manner. For many applications, however, this may indeed by the best approach in spite of its drawbacks. There is certainly nothing wrong with such a "brute force" approach so long as it totally fulfills the needs of the user.

The second approach is slightly more complicated and definitely much more expensive than the basic interface, but produces some very useful "spin-offs". This is done by using an FCC approved data coupler, also commonly called a Data Access Arrangement (DAA). There are three basic types of these devices. The first, a CDT data coupler, is not suitable for this application since it has control over the voice (or data) mode of a telephone. The other types CBS and CBT have control over all functions of the telephone lines, including dialing and answering.

The DAA serves many functions because it is designed to be an interface between a direct connect modem and the telephone line. Not only does it connect a terminal/modem device to the telephone lines by an isolation transformer, but it has to have circuitry for limiting the signal level going over thetelephone lines, thereby limiting the bandwidth of the signal and assuring proper impedance to the lines. In addition, there are relays and circuitry for ring detection, switch hook control and other line functions. These additional functions are there for sophisticated terminal equiptment which have, among other things (you guessed it) the capability of auto dialing. The price of a CBT type datacoupler is about $125—$200, depending on the source, and these are sold mostly by companies that also sell modems. In some areas, they can be rented from the telephone company for about $5 to $7 per month. Used CBTs when available, sell for about $80—$125.

At this point, one immediately says, "Wait a minute! Even if I can get a used CBT, I'm approaching the cost of an off-the-shelf auto-dialer. What am I gaining here?" Obviously, one still has a much more versatile system than these dialers. But here

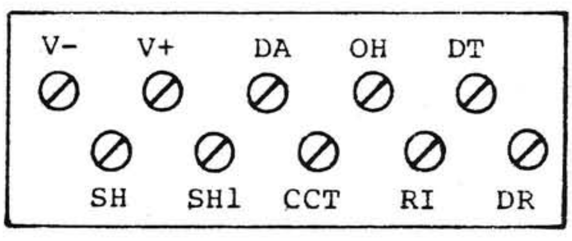


Figure 2: **CBT Interface Connections**

is where some "spin-off" occurs. Ever since the recent FCC part 68 ruling which allows, among other things, for modem manufacturers to produce FCC approved equipment without having to go through a separate DAA, high quality used modems are coming onto the market at real bargain prices. Also, there are available numerous single board modems which plug directly into S-100 and other bus systems and terminal equiptment. What all but a few advertisers of both of these equipment fail to mention is that to legally install them (unless the device is FCC approved), one must have an FCC approved DAA! Therefore, if the purchase of a data communication system is being contemplated, consider this. With judicious shopping for a used high quality direct connect modem, a DAA and using the SYM dialer as a controller, one can come up with a sophisticated data communications system capable of auto dialing, auto answer, auto originate and a few other bells and whistles. Such a system could cost up to $2000, but by going this route, it can be obtained at a fraction of this cost.

Now to the actual interfacing itself. Either a CBS or CBT type coupler could be used, but the better choice here is the CBT. The CBS is more expensive and uses RS-232 signal levels for control which would add componets and complexity to the interface. The CBT uses a switch closure concept for its operation.

Figure 2 shows the user end connections to the CBT. For the purpose of dialing, the only ones of interest here are the V- and OH (Off Hook) connections. When V- and OH are connected this causes the line leads to become active (as is taking the receiver Off Hook), and the CBT takes over control of the line. Therefore, a number can be dialed by pulsing a connection between these two leads. This may be accomplished with the contacts of the relays in Figure 2 connected between the V- and OH connections. Notice what has happened by this revision of the circuit—essentially nothing! The combination of the relay and CBT have exactly the same net effect as the original configuration connected directly to the line. Therefore, the same driving

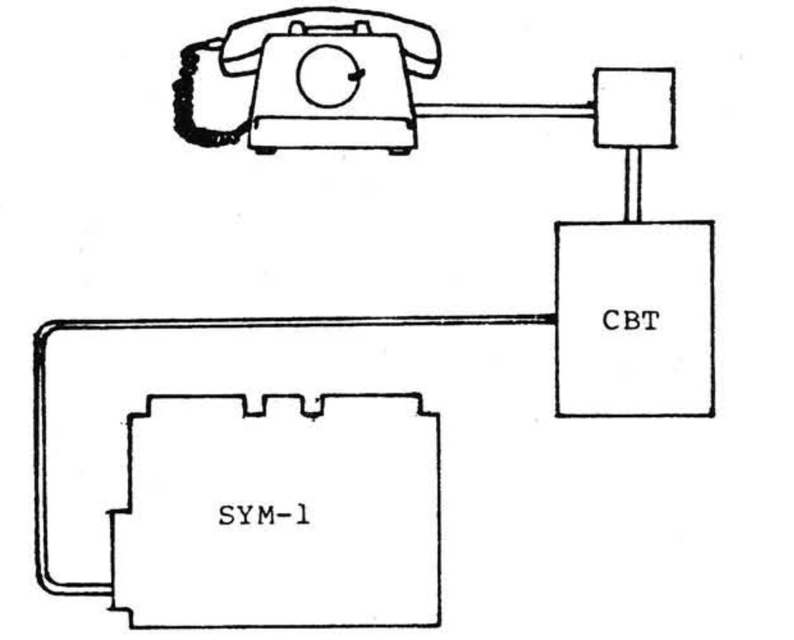


***Figure 3:* The "Legal Connection"**

software can be used, with one exception. In using the basic interface, it has been assumed that the initial off hook condition is done manually after the directory and instructions have been printed out. This is not the case for the CBT, since the initialization call at BASIC program line number 160 now causes an off hook condition, and capture of the telephone line by the DAA. In the ensuing time used for the printing, some telephone's systems may "time out" and issue a whining sound to indicate that the central office has deactivated the telephone line. The telephone must be placed back on hook before normal operation can be continued. Therefore, BASIC line 160 should be changed to line 525 to avoid this potential problem.

The CBT has a modular plug which merely plugs into the wall. In order to plug the telephone and the CBT into the same wall jack a device called a duplex adapter may be used. This is a small device which plugs into a modular jack and allows two plugs to go into it. This type of device can be found in most stores that sell telephone accessories. Its primary use is to allow connection of a telephone to the same jack as an answering machine, hands-off amplifier and would you believe some types of auto dialers? It should be noted at this time that some DAA's come with a special plug intended for use with a data jack. This is an eight pin modular plug instead of the usual six pin voice plug.

In this case a special adaptor to go from the eight pin to six pin plug must be used. Figure 3 shows a pictorial hookup of the "Legal" Connection.

The above discussion on CBT type data couplers, is at best, cursorary. A full discussion on DAA's is enough for quite a lengthy article in itself. Enough information has been presented in order to pulse dial a telephone using this device. Although all data couplers must be, by virture of FCC standards, functionally equivalent; there are some differences between manufacturers as to how these functions are accomplished internally. Therefore, the user's manual for the particular data coupler to be used should be read throughly before working on this interface.

System Checkout

Before actually using the interface and driving programs, they should be thoroughly checked out. You don't really want to have to pay for a phone call to Pago-Pago, do you? A good way to do this is to use an LED and series resistor instead of the relay. At lines 119 and 125 of the machine language routines, change the constant "01" to "0A". This will result in a 1 Hz rate of dialing instead of 10 Hz. The individual pulses can then be easily counted and the operation of the interface can be monitored. Then, when everything is working properly, connect the relay.

Other Applications

There are many changes which could be made to the presented BASIC program to alter or enhance its dialing capability. For instance, instead of entering a directory number, the program could be changed to accept a directory name. Another possibility is to store a table of basic telephone rates for each number, and by use of the timer, the cost of any call could be automatically computed and printed out. Also, dialing a number not in the directory by keying in the number would still be faster for the user than using the telephone's dialer. For the aid of sightless persons, how about a voice interface to trigger dialing? As you can see, the variations are limited only by the imagination.

Randy Sebra received his BS degree in Physics fron Virginia Polytechnic Institute and State University in 1966, and currently works as an operations research analyst for the United States Army. Performing his duties in the analysis of weapons systems, he relies quite heavily on the use of computers.

He tells us, "Experimenting with my SYM-1 at home is not quite the 'busman's holiday' that it may appear at first. In my work I am not able to get into too much hardware, being restricted mostly to digital simulation and mathematical modeling. My SYM-1 gives me the opportunity to get my hands 'dirty' with hardware and interfacing."

Listing 1

```
 1:                       Machine language routines used with
 2:                       SYM-1 Telephone Dialer.  Written by
 3:                       Randy Sebra , APRIL, 1980.
 4:
 5:                          OUTBYT  EQU  $82FA     Monitor, output HEX byte
 6:                          SCAND   EQU  $8906     Monitor, scan LED display
 7:                          OUTCHR  EQU  $8A47     Monitor, output ASCII character
 8:                          ACCESS  EQU  $8B86     Monitor, un-write protect RAM
 9:                          PDBA    EQU  $A402     Output Register B, system 6532
10:                          OUTVEC  EQU  $A664     Output Vector
11:                          IRQVEC  EQU  $A67E     IRQ Vector
12:                          ORB     EQU  $AC00     Output Register B
13:                          T1LL2   EQU  $AC04     Timer 1 Low Counter(read)
14:                          T1LH    EQU  $AC05     Timer 1 High Counter(write)
15:                          T1LL    EQU  $AC06     Timer 1 Low Latch
16:                          T2LL    EQU  $AC08     Timer 2 Low Latch
17:                          T2CH    EQU  $AC09     Timer 2 High Counter
18:                          ACR     EQU  $AC0B     Aux. Control Register
19:                          IFR     EQU  $AC0D     Interrupt Flag Register
20:                          IER     EQU  $AC0E     Interrupt Enable Register
21:
22:                                  ORG  $0F1A    Start of Routines
23:
24:                            Driving Routine for timer
25:
26:0F1A-   20 86 8B   TIMER   JSR   ACCESS     Un-write protect system RAM
27:0F1D-   A9 6F              LDA   $6F        Change IRQVEC Vector to
28:0F1F-   8D 7E A6           STA   IRQVEC     interrupt routine at
29:0F22-   A9 0F              LDA   $0F        $0F6F
30:0F24-   8D 7F A6           STA   IRQVEC+1
31:0F27-   A9 EC              LDA   $EC        Load loop counter
32:0F29-   85 F0              STA   COUNT
33:0F2B-   A9 00              LDA   $00        Initialize minutes and
34:0F2D-   85 F1              STA   SEC        seconds count
35:0F2F-   85 F2              STA   MIN
36:0F31-   A9 40              LDA   $40        Set ACR for timer 1
37:0F33-   8D 0B AC           STA   ACR        free running mode
38:0F36-   A9 C0              LDA   $C0        Enable interrupt
39:0F38-   8D 0E AC           STA   IER
40:0F3B-   A9 4F              LDA   $4F        Set up timer 1 for
41:0F3D-   8D 06 AC           STA   T1LL       a .05 sec delay
42:0F40-   A9 C3              LDA   $C3        and start timer
43:0F42-   8D 05 AC           STA   T1LH
44:0F45-   A9 00              LDA   $00        Change OUTVEC so
45:0F47-   8D 64 A6           STA   OUTVEC     that the output goes
46:0F4A-   A9 89              LDA   $89        to the LED display.
47:0F4C-   8D 65 A6           STA   OUTVEC+1
48:0F4F-   20 93 0F           JSR   DSPLAY     Initialize display
49:0F52-   20 06 89   SCAN    JSR   SCAND      Scan the display
50:0F55-   AD 02 A4           LDA   PDBA       Check for terminal key down
51:0F58-   10 F8              BPL   SCAN       Scan until detected
52:0F5A-   A9 40              LDA   $40        Disable interrupt
53:0F5C-   8D 0E AC           STA   IER
54:0F5F-   A9 A0              LDA   $A0        Change OUTVEC back to
55:0F61-   8D 64 A6           STA   OUTVEC     terminal
56:0F64-   A9 8A              LDA   $8A
57:0F66-   8D 65 A6           STA   OUTVEC+1
58:0F69-   4C B6 0F           JMP   HANG       Do a hang up
59:0F6C-   4C 4C D1           JMP   D14C       Return to BASIC
60:
61:                      Interrupt Routine - updates elapsed time
62:
63:0F6F-   48         UPDATE  PHA              Save accumulator on stack
64:0F70-   18                 CLC              Clear carry for add
65:0F71-   E6 F0              INC   COUNT      Increment loop counter
66:0F73-   D0 19              BNE   OUT2       If a second not up, skip
67:0F75-   F8                 SED              Else set decimal mode
68:0F76-   A9 EC              LDA   $EC
69:0F78-   85 F0              STA   COUNT      Re-set loop counter
70:0F7A-   A5 F1              LDA   SEC        Increment seconds count
71:0F7C-   69 01              ADC   $01
72:0F7E-   85 F1              STA   SEC        and re-store
73:0F80-   C9 60              CMP   $60        A minute up ?
74:0F82-   D0 06              BNE   OUT1       If not, skip
75:0F84-   A9 00              LDA   $00        Else, re-set seconds
```

```
0F86-   85 F1              STA   SEC       count and increment
0F88-   E6 F2              INC   MIN       minutes count
0F8A-   D8        OUT1     CLD             Clear decimal mode
0F8B-   20 93 0F           JSR   DSPLAY    Display elapsed time
0F8E-   AD 04 AC  OUT2     LDA   T1LL2     Clear interrupt flag
0F91-   68                 PLA             Restore accumulator
0F92-   40                 RTI             Return from interrupt

                  Display Routine

0F93-   A9 20     DSPLAY   LDA   $20       Write out a space
0F95-   20 47 8A           JSR   OUTCHR
0F98-   A5 F2              LDA   MIN       Write out minutes
0F9A-   20 FA 82           JSR   OUTBYT
0F9D-   A9 20              LDA   $20       Write out a space
0F9F-   20 47 8A           JSR   OUTCHR
0FA2-   A5 F1              LDA   SEC       Write out seconds
0FA4-   20 FA 82           JSR   OUTBYT
0FA7-   60                 RTS

                  Initialization Routine

0FA8-   20 86 8B  INIT     JSR   ACCESS    Un-write protect RAM
0FAB-   A9 80              LDA   $80       Configure PB7 as
0FAD-   8D 02 AC           STA   DDRB      an output port
0FB0-   A9 00     PBOFF    LDA   $00       and turn it off
0FB2-   8D 00 AC           STA   ORB
0FB5-   60                 RTS

                  Hang up Routine

0FB6-   A9 80     HANG     LDA   $80       Turn PB7 on
0FB8-   8D 00 AC           STA   ORB
0FBB-   20 E4 0F           JSR   DPAUSE    Do a 800 msec delay
0FBE-   4C B0 0F           JMP   PBOFF     Turn off PB7

                  Dialing Routine - Enter with the number
                  of pulses to be dialed in accumulator.

0FC1-   AA        DIAL     TAX             Transfer # of pulses to Xr
0FC2-   48                 PHA             and save on stack
0FC3-   A9 80              LDA   $80       Turn PB7
0FC5-   8D 00 AC           STA   ORB       on
0FC8-   A2 01              LDX   $01       # of times through DELAY=1
0FCA-   A0 ED              LDY   $ED       Set up timer 2 for a
0FCC-   98                 TYA             .061 sec delay
0FCD-   20 EE 0F           JSR   DELAY     Do the delay
0FD0-   A9 00              LDA   $00       Turn PB7
0FD2-   8D 00 AC           STA   ORB       off
0FD5-   A2 01              LDX   $01       # of times through DELAY=1
0FD7-   A0 86              LDY   $86       Set up timer 2 for a
0FD9-   A9 9E              LDA   $9E       .039 sec delay
0FDB-   20 EE 0F           JSR   DELAY     Do the delay
0FDE-   68                 PLA             Restore the # of
0FDF-   AA                 TAX             pulses counter and
0FE0-   CA                 DEX             decrement it
0FE1-   8A                 TXA             Transfer counter back to Ar
0FE2-   D0 DE              BNE   DIAL+1    Loop for proper # of pulses
0FE4-   A2 10     DPAUSE   LDX   $10       # of times through DELAY=16
0FE6-   A0 C3              LDY   $C3       Set up timer 2 for a .05
0FE8-   A9 4F              LDA   $4F       sec delay, total= .8 sec
0FEA-   20 EE 0F           JSR   DELAY     Do the delay
0FED-   60                 RTS             Return

                  General Delay Routine - Enter with number
                  of times through in the X register, low
                  and high bytes for timer in A,Y register pair.

0FEE-   8D 08 AC  DELAY    STA   T2LL      Write to low order latch
0FF1-   98                 TYA
0FF2-   8D 09 AC           STA   T2CH      Write to high order counter
0FF5-   AD 0D AC  CHECK    LDA   IFR       Check interrupt flag
0FF8-   29 20              AND   $20       register for time-out
0FFA-   F0 F9              BEQ   CHECK     If not, loop until it has
0FFC-   CA                 DEX             Decrement times through count
0FFD-   D0 F2              BNE   DELAY+3   Loop until through
0FFF-   60                 RTS
```

Listing 2

```
100 REM      **  TELEPHONE DIALER
110 REM      **  RANDY SEBRA
120 REM      **  APRIL,1980
130 REM
140 REM      **  INITIALIZE INTERFACE
150 REM
160 Z=USR(&"0FA8",0)
170 DIM N$(19),T$(19)
180 DATA POLICE,FIRE,DOCTOR,LAWYER,SCHOOL,PARENTS,WORK
190 DATA WIFE'S WORK,NEIGHBOR,BROTHER,JANE,JOE,JOHN,SALLY,JIM
200 DATA JOAN,DORIS,BILL,HOME,COMPUTER
210 DATA 555-0000,555-3333,555-5894,555-3958,555-5683,1-804-559-6741
220 DATA 1-557-9338,1-557-4736,1-557-9939,1-703-556-0924,555-0226
230 DATA 555-9328,555-1293,555-3092,555-8876,555-2783,555-5638,555-9951
240 DATA 9.555-4702,555-4900.554-1200
250 REM
260 REM      **  READ AND PRINT OUT DIRECTORY
270 REM
280 FOR I=0 TO 19 STEP 2
290 READ N$(I),N$(I+1)
300 PRINT I;TAB(5);N$(I);TAB(20);I+1;TAB(25);N$(I+1)
310 NEXT I
320 REM      **  READ NUMBERS
330 FOR I=0 TO 19
340 READ T$(I)
350 NEXT I
360 PRINT
370 REM
380 REM      **  PRINT INSTRUCTIONS
390 REM
400 PRINT "FIRST PICK UP RECEIVER AND WAIT FOR DIAL TONE."
410 PRINT "ENTER THE DIRECTORY NUMBER(S) YOU WISH TO DIAL. YOU MAY"
420 PRINT "ENTER A SINGLE NUMBER, A SEQUENCE OF NON-CONSECUTIVE"
430 PRINT "NUMBERS SEPARATED BY SEMI-COLONS, OR A RANGE OF"
440 PRINT "NUMBERS SEPARATED BY A DASH."
450 PRINT
460 PRINT "ANY TIME YOU WISH TO HANG UP, ENTER AN H. TO RE-DIAL THE"
470 PRINT "PREVIOUS NUMBER, ENTER AN R(HANG UP NOT NECESSARY). TO"
480 PRINT "CONTINUE AFTER AN ACCESS PAUSE, ENTER C(OR H OR R IF THE"
490 PRINT "LINE IS BUSY). TO USE TIMER, ENTER A T AFTER THE"
500 PRINT "CALL IS ANSWERED. WHEN THE CONVERSATION IS OVER,"
510 PRINT "PRESS ANY KEY TO STOP TIMER AND HANG UP."
520 PRINT
530 INPUT "READY  ";Z$
540 S=VAL(Z$)
550 B$=STR$(S)
560 L1=LEN(Z$)
570 L2=LEN(B$)-1
580 GOSUB 740
590 REM      **  SINGLE NUMBER
600 IF L1=L2 THEN 700
610 IF MID$(Z$,L2+1,1)<>";" THEN 650
620 REM      **  NON-CONSECUTIVE SEQUENCE
630 Z$=MID$(Z$,L2+2)
640 GOTO 540
650 IF MID$(Z$,L2+1,1)<>"-" THEN STOP
660 REM      **  CONSECUTIVE SEQUENCE
670 S=S+1
680 GOSUB 740
690 IF S<VAL(MID$(Z$,L2+2)) THEN 670
700 INPUT "RUN AGAIN(Y OR N) ";Z$
710 IFZ$="Y" THEN 530
720 END
730 REM      **  DIALING ROUTINE
740 PRINT "DIALING  ";N$(S);" ";
750 FOR I=1 TO LEN(T$(S))
760 A$=MID$(T$(S),I,1)
770 IF A$="-" THEN 850
780 IF A$="0" THEN A$="10"
790 IF A$<>"." THEN 830
800 INPUT "ACCESS PAUSE - USE C,H OR R OPTIONS.";Y$
810 IF Y$<>"C" THEN 950
820 GOTO 860
830 A%=VAL(A$)*256
840 D=USR(&"0FC1",A%)
850 PRINT RIGHT$(A$,1);
860 NEXT I
870 INPUT Y$
880 IF Y$<>"T" THEN 950
890 T=USR(&"0F1A",0)
900 S1=PEEK(241)-INT(PEEK(241)/16)*6
910 M1=PEEK(242)-INT(PEEK(242)/16)*6
920 PRINT
930 PRINT "ELAPSED TIME :";M1;"MINUTES AND";S1;"SECONDS"
940 RETURN
950 IF Y$<>"H" THEN 980
960 H=USR(&"0FB6",0)
970 RETURN
980 IF Y$<>"R" THEN STOP
990 R=USR(&"0FB6",0)
1000 GOTO 740
1010 RETURN
```

μ

Compare Our Prices With Any Others

Rockwell's	AIM-65	1K System:	**$405.**	4K System	**$459.**
Synertek's	SYM-1	1K System:	**235.**	4K System	**259.**
Commodore's	KIM-1	1K System:	**175.**		

FOR YOUR SYSTEM'S EXPANSION

The Computerist, Inc's:

16K DRAM	**$279.**	Proto Plus II	**$42.**
32K DRAM	**375.**	ASK I/O Board	**55.**
Video Plus II	**279.**	DRAM & Video Cable	**15.**
Mother Plus II & Card Cage	**115.**	Power Supply for SYM-1	**39.**

Power Supply and Enclosure for AIM-65	**$119.**
Power Supply and Enclosure for KIM-1	**65.**

All products are factory warrantied. Prices include full documentation.

Send Check or Money Order to:

Hepburn MCA*

12 Grosvenor Street

Lowell, MA 01851

Please add $5.00 shipping and handling. MA residents add 5% sales tax.

** Mini Computers and Accessories*

WANT YOUR COMPUTER BACK?

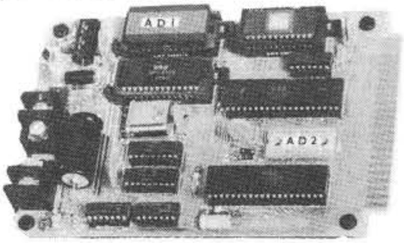
Let the MICROsport Microcomputer (MMC) take over any dedicated task.

It is the affordable alternative (kits from $89.00), application units, A & T, from only $119.00).

It is user-oriented – complete in-circuit emulation allows program development on ANY 6502 based system. It is compact (4½" by 6½" pc board) but powerful (32 I/O 6503 CPU) and works off any AC or DC power supply.

Turn your present 6502 based system into a complete development system with:

1 MMC/03D Microcomputer with ZIF sockets
1 MMC/03ICE In-circuit emulator for the 6503 CPU
1 MMC/03EPA EPROM Programmer complete with software driver.



All for only $250.00! (Offer good until December 1980.)

For more infor call or write

R. J. BRACHMAN ASSOCIATES, INC.
P.O. Box 1077
Havertown, PA 19083
(215) 622-5495

PET and APPLE II Users

PASCAL

ABACUS Software makes available its version of TINY PASCAL for the users of two of the most popular personal computers.

TINY PASCAL is a subset of the standard PASCAL as defined by Jensen and Wirth. It includes the structured programming features: IF-THEN-ELSE, REPEAT-UNTIL, FOR TO/DOWN TO-DO, WHILE-DO, CASE-OF-ELSE, FUNC and PROC. Now you can learn the language that is slated to become the successor to BASIC.

TINY PASCAL is a complete package that allows you to create, compile and execute progams written in the PASCAL language. You can save source and object code on diskette or cassette (PET version only). Comprehensive user's manual included. The manual can be examined for $10 (refundable with software order).

REQUIREMENTS

PET 16K/32K New ROMS cassette	$40
PET 16K/32K New ROMS diskette	$35
Apple II 32K Applesoft ROM w/DOS	$35
Apple II 48K Applesoft RAM w/DOS	$35
TINY PASCAL User's Manual	$10
6502 Interpreter Listing	$20

FREE postage in U.S. and CANADA
All orders prepaid or COD

ABACUS SOFTWARE
P. O. Box 7211
Grand Rapids, Michigan 49510

PET Vet

by Loren Wright
PET Specialist
MICRO Staff

On September 22 I took the Amtrak "Night Owl" to Washington, D.C., and I returned on the "Night Owl" the night of the 24th. While there, I attended the Federal Computer Conference at the Sheraton Washington. This conference is aimed at the many government agencies that have occasion to use computers in their work. All of the "biggies" had displays there, but Commodore was the only 6502 microcomputer manufacturer to have a booth. CBM's and PET's were busy demonstrating different business software packages.

On the second day, I had a long talk with Commodore's new Manager of Public Relations, G. Thomas Sheffer. Within the next few months he plans to mail a questionnaire to all PET Users' Club members in order to help determine the future direction and content of the Commodore *Newsletter*. The Users' Club and its *Newsletter*, now the responsibility of Public Relations, should be a reliable source of information for PET owners (Subscription - $15; 10 back issues - $15).

Editor, Commodore Newsletter
Public Relations Department
Commodore Business Machines
950 Rittenhouse Road
Norristown, PA 19403

The Transactor, from Commodore in Canada, has long been a valuable information source. Subscriptions start with the beginning of a volume only. The current volume is II, but is nearly completed. Volume I and II are each $15.

Editor, The Transactor
Commodore Systems
3370 Pharmacy Avenue
Agincourt, Ontario, Canada

Commodore Product Summary

Commodore sells a wide line of computer products, but even PET owners may be a little confused by all the different model numbers. Starting this month with the computers themselves, I will try to explain the differences. Next month I will cover the peripherals. The new CBM 4040 dual-floppy drive was exhibited at the Federal Computer Conference and will be generally available in late November.

When owners of the other home computers think of the PET, they think of what is now called the PET 2001-8KS. Although this has been out of production since January, many MICRO readers have them, and are very happy with them. These have a small (calculator style) keyboard, an integral cassette recorder, and 2.0 BASIC. The keyboard was difficult for most people to use, and tended to develop reliability problems. I didn't object to the close spacing of the keys, since I have skinny fingers, but I was occasionally frustrated by keys that wouldn't register or ones that "bounced." These are still available, both new and used, at very reasonable prices. Upgrade BASIC ROM kits and full-size expansion keyboards can be obtained.

Commodore currently makes three lines of computers: the PET 2001 series, the CBM 2001 series, and the CBM 8000 series. Both 2001 series contain the 3.0 BASIC, and the 8000's contain 4.0 BASIC.

The principal difference between the PET 2001 and CBM 2001 lines is in the keyboards. PET keyboards are called *graphics* keyboards because, in addition to letters and numbers printed on the key tops, graphics characters are printed on the key fronts. The number keys are in a separate keypad to the right, along with cursor movement keys and the period. Characters used frequently in entering a BASIC program, such as **: ? $ %** and **#,** can be typed without shifting. Capital letters and numbers are entered without shifting. When the shift key is pressed, all the graphics characters are available.

PET's and CBM's have two character sets, of which only one can be displayed at any given time. One includes all the graphics characters. The other substitutes lower case letters for those graphics appearing on the letter keys. This means that in order to get lower case letters in this character set, the shift key must be pressed for each—the reverse of normal typewriter operation. When PET's are powered up they are in the graphics character set, and to switch to lower-case character set, the statement "POKE 59468,14" must be executed.

Current production PET models are listed with model numbers and list prices. The **N** suffix indicates the *graphics* keyboard. **8, 16,** and **32** indicate the number of kilobytes of RAM included.

PET 2001- 8N	$ 795
PET 2001-16N	$ 995
PET 2001-32N	$1295

Models in the CBM-2001 line have the *business* keyboard. This is very similar to a standard typewriter keyboard. When powered-up, all letters are lower case, and their shifts are upper case. Numbers appear in their standard positions above the letters, as well as in the separate numeric keypad. Characters such as **! # $ %** and **:**, must be shifted, but the period is in its normal position below the **L**. "POKE 59468, 12" must be executed in order to utilize the graphics character set, and the characters must be looked up in a table, since they don't appear on the keys. This character set configuration and keyboard layout are particularly well suited to word processing and other business applications.

CBM 2001-16B	$ 995	*16K RAM*
CBM 2001-32B	$1295	*32K RAM*

The CBM 8000 series computers have several differences. 4.0 BASIC differs from 3.0 BASIC primarily in the addition of several disk commands, which make communication with DOS 2.1 a lot easier. The screen is 80 characters wide, and physically larger as well. The keyboard is a business-style keyboard, but with several keys added and others relocated. The advantages of the 8000 series machines for business applications such as ledger and word-processing, should not be overlooked.

CBM 8016	$1495	16K RAM
CBM 8032	$1795	32K RAM

Self-modifying PET Programs

Use this simple tutorial to write a self-modifying program.

P. Kenneth Morse
P.O. Box 3367
Augusta, GA 30902

High-level languages, such as BASIC, typically make it difficult or impossible to modify the program itself as a result of the program's own operation. There is good reason for this, since such changes are usually difficult to detect and debug when they occur inadverently. Nevertheless, there are times when we might wish to develop programs that do modify themselves. Since "old" 8K PET casette data files are somewhat unreliable, due to bugs in the operating system, one reasonable application would be to generate (or delete) DATA statements under program control, thus capitalizing on the greater reliability of program tapes.

The methodology for accomplishing this was explained by Mike Louder in "The PET Has a Dynamic Keyboard" (*PET User Notes,* 1978, 1, issue 6, p. 11). The methodology capitalizes on the fact that when a program terminates execution with an END statement, location 525 (158 for "new" ROMs) is checked to see if any unexecuted instructions are in the keyboard buffer (locations 527—536 for old ROMs, 623—632 for new). If so, it executes them. Now, if we could LOAD the keyboard buffer before exiting the program, those instructions would be carried out after the program was over. And if the last instruction were to cause the program to re-start...well, we could then write programs that would modify themselves and continue to run! As it turns out, we can do just that! Here are four projects to help you learn the technique and its limits.

Project 1

The variable PT may be a bit puzzling. Since location 50003 always has the value "0" with the "old"ROMs and "1" with the 'new' ROMs, we can use

PT = PEEK(50003)

to adjust addresses automatically, using the formula

(ADDRESS) = (OLD ADDRESS) + PT*(adjustment factor)

Whenever PT = 0 (old ROMs), the adjustment vanishes, since zero times anything is zero.

```
10  GO TO 50
20  READ N$
30  PRINT "THE NAME IS ";N$
40  STOP
50  INPUT "YOUR NAME, PLEASE ";A$
60  PRINT "cdddl000DATA ";A$
70  PRINT "GOTO 20"; "h"
80  J=1
90  REM: LOAD KEYBOARD BUFFER WITH
                          'RETURNS'
100 PT=PEEK (50003)
110 POKE 525-PT*367,J+1
120 FOR K=1 TO J + 1
130 POKE 526 + K + PT*96, 13
140 NEXT K
150 END
```

Note: lower case letters in quotes stand for special PET keys:

"c" = clear screen
"d" = cursor down
"h" = home

1. RUN this program by entering the name "JOHN DOE". The results should be:

```
READY.
1000DATA JOHN DOE
GOTO 20
THE NAME IS JOHN DOE

BREAK IN 40
READY.
```

2. LIST this program (after RUNning it), and you will find a new line:

```
1000 DATA JOHN DOE
```

3. Now, enter the immediate command

```
?N$
```

then

```
?A$
```

Note that A$ has been lost! One complication with this technique is that the program re-initializes all variables when it re-starts by executing the on-screen command. Hence, A$ is now equal to " ". There are two ways to handle this problem: one is to record the value of the variable in a new (or altered) DATA statement, as was done above. The other way is shown in project 2.

Project 2

Make the following changes:

```
1000 (deleting the DATA statement)
  10 Q$=CHR$(34):GOTO 50
  20 REM
  60 PRINT "cdddN$=";Q$;A$;Q$
```

1. RUN: how does the result compare with Project 1?

2. LIST: note that no DATA statement is present. Yet, the PRINT statement in line 30 was able to recognize as N$ the name originally entered as A$.

There is one important point to watch. Several DATA statements may be generated with a single program exit, but only a single line (up to 40 characters) of direct command may be entered.

We are now beginning to identify some rules for self-modifying programs. Before exiting, the program should:

1. Clear the screen.

2. PRINT the BASIC lines to be incorporated into the program on the screen, beginning with the fourth line from the top. Each BASIC line may be up to 78 characters long, and should be single-spaced.

3. Following the BASIC lines, PRINT a single unnumbered line (no more than 40 characters) containing any variables that need to be saved to restore the program to the same point of operation. End the line with a GOTO statement returning control to the main program (*not* to a subroutine).

4. POKE the value of N (where N = number of BASIC lines + 1) into the keyboard index byte, and POKE the value "13" into each of N bytes in the keyboard buffer.

5. "Home" the cursor.

6. Exit from the program with an END statement.

Project 3

How many BASIC lines can be created under program control with a single program exit? Make these changes in your program:

```
 7Ø (delete)
 8Ø (delete)
 1Ø INPUT "VALUE OF J ";J
 2Ø PRINT "cddd";
 3Ø FOR I = 1 TO J
 4Ø PRINT I*1ØØØ; "DATA ";I;I*I;SQR(I)
 5Ø NEXT I
 6Ø PRINT "LIST"
15Ø PRINT "h"
16Ø END
```

Project 1

```
 10 GOTO 50
 20 READ N$
 30 PRINT "THE NAME IS ";N$
 40 STOP
 50 INPUT "YOUR NAME, PLEASE ";A$
 60 PRINT "cddd1000DATA ";A$
 70 PRINT "GOTO 20";"h"
 80 J=1
 90 REM: LOAD KEYBOARD BUFFER WITH 'RETURNS'
 100 PT=PEEK(50003)
 110 POKE 525-PT*367,J+1
 120 FOR K=1 TO J+1
 130 POKE 526+K+PT*96,13
 140 NEXT K
 150 END
READY.

READY.
PROJECT 1
READY.
```

Project 2

```
 10 Q$=CHR$(34): GOTO 50
 20 REM
 30 PRINT "THE NAME IS ";N$
 40 STOP
 50 INPUT "YOUR NAME, PLEASE ";A$
 60 PRINT "cdddN$="; Q$; A$; Q$
 70 PRINT "GOTO 20";"h"
 80 J=1
 90 REM: LOAD KEYBOARD BUFFER WITH 'RETURNS'
 100 PT=PEEK(50003)
 110 POKE 525-PT*367,J+1
 120 FOR K=1 TO J+1
 130 POKE 526+K+PT*96,13
 140 NEXT K
 150 END
READY.
```

Project 3

```
 10 INPUT "VALUE OF J ";J
 20 PRINT "cddd";
 30 FOR I=1 TO J
 40 PRINT I*1000; "DATA "; I; I*I; SQR(I)
 50 NEXT I
 60 PRINT "LIST"
 90 REM: LOAD KEYBOARD BUFFER WITH 'RETURNS'
 100 PT=PEEK(50003)
 110 POKE 525-PT*367,J+1
 120 FOR K=1 TO J+1
 130 POKE 526+K+PT*96,13
 140 NEXT K
 150 PRINT "h"
 160 END
READY.

READY.
```

SAVE project 3 on tape (and VERIFY) before proceeding.

Begin with a value of J = 1 and continue, i ncreasing by 1 each time, until you "crash" BASIC or get an error message. When this happens, you know you have one line too many! Each time, the program will LIST its current version. Note how many DATA statements were created each time. To be sure that the program is generating all of the DATA statements each time, type

NEW

and re-enter the original program from tape. Then, RUN, and increase the value of J by 1. (*Note:* by deleting line 60 and changing "J + 1" in lines 110 and 120 to "J", the maximum number of DATA statements generated can be one greater, but then no immediate commands ... such as LIST or GOTO 10...may be provided under program control.)

Project 4

```
10 INPUT "VALUE OF J ";J
20 PRINT "[illegible]";
30 FOR I=1 TO J
40 PRINT I*1000
50 NEXT I
60 PRINT "LIST"
90 REM: LOAD KEYBOARD BUFFER
            WITH 'RETURNS'
100 PT=PEEK(50003)
110 POKE 525-PT*367,J+1
120 FOR K=1 TO J+1
130 POKE 526+K+PT*96,13
140 NEXT K
150 PRINT "[illegible]"
160 END
1000 DATA  1  1  1
2000 DATA  2  4  1.41421356
3000 DATA  3  9  1.73205081
4000 DATA  4  16  2
5000 DATA  5  25  2.23606798
READY.
```

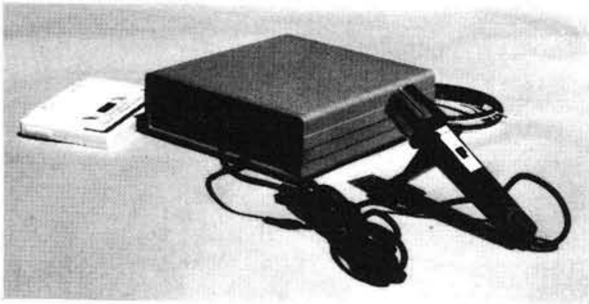
Project 4

How about deleting lines under program control? Make one change:

40 PRINT I*1000

SAVE the latest version of the program (including all the DATA statements) as project 4 and VERIFY. RUN the program. When the program LISTs itself, you will note that some or all of the DATA statements (depending on the value of J) will have disappeared. Since you SAVEd the set of DATA statements, you can experiment with this program at your leisure.

SPEECH RECOGNITION



NOW TALK TO YOUR:

PET, TRS-80 (LII) AIM-65 and SORCERER

NEW COGNIVOX SR-100 has 32 word (or short phrase) vocabulary (AIM-65 with 4K RAM, 16 words). Up to 98% recognition rate. Breakthrough price of only **$119** includes microphone, cassette with software and manual. Version for the TRS-80 (VIO-332) costs $149 but also has 32 word speech output and music capability, includes all above plus speaker/amplifier. For the Sorcerer, in addition to SR-100, we offer COGNIVOX VIO-132 which includes speech output and music and extensive software and costs $179. Please add $3 for shipping in the US. Calif. add 6% tax. Foreign orders welcome, add 10% for processing and shipping by air. When ordering, please specify make, model and memory size of your computer.

VOICETEK

Dept. **M**, *P.O. Box 388, Goleta, Ca 93017*

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 5' 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

APPLE & PET

MAE

The Most Powerful Disk-Based Macro Assembler/Text Editor Available at ANY Price

Now includes the Simplified Text Processor (STP)

For 32K PET, disk 3.0 or 4.0 ROMS or 8032 (specify) — OR — 48K APPLE II or APPLE II + and DISK II

MAE FEATURES
- Control Files for Assembling Multiple named source files from disk
- Sorted Symbol table — Up to 31 chars./label
- 27 Commands, 26 Pseudo-ops, 39 Error Codes
- Macros, Conditional Assembly, and a new feature we developed called Interactive Assembly
- Relocatable Object Code
- String search and replace, move, copy, automatic line numbering, etc.

STP FEATURES
- 17 text processing macros
- Right and left justification
- Variable page lengths and widths
- Document size limited only by disk capacity
- Software lower case provision for APPLE II without lower case modification

ALSO INCLUDED
- Relocating Loader
- Sweet 16 macro library for APPLE and PET
- Machine Language macro library
- Sample files for Assembly and text processing
- Separate manuals for both APPLE and PET

PRICE
- MAE, STP, Relocating Loader, Library files, 50 page manual, diskette — $169.95

SEND FOR FREE DETAILED SPEC SHEET

EASTERN HOUSE SOFTWARE
3239 LINDA DRIVE
WINSTON-SALEM, N. C. 27106

(919) 924-2889 (919) 748-8446

OHIO SCIENTIFIC

SOFTWARE

COPY/1 CAN COPY A FULL DISK IN UNDER TWO MINUTES, USING A SINGLE DRIVE, FROM A COLD START.

No messing around with disk initializing or track zero set up or directory scan, these are all handled by COPY/1. COPY/1 needs 24k RAM; it can make 0 to 255 copies and can start and stop on any tracks. It lists the sector page count following each sector read or write action. It lists disk error type and location. All in full color with sound cues. (Also works fully without color or sound).

Cost $20 (plus 5% tax, Md. residents) includes disk and mailing.

From
PRISM SOFTWARE
Box 928
College Park, Md.
20740

Prism Software

MICRO SCOPE

Number 4

General Ledger for the APPLE II

$180.00

(+$20.00 for Osborne Associates' book: *General Ledger in C-Basic.*)

Small Business Computer Systems (SBCS)

4140 Greenwood
Lincoln, NE 68504

1-3. Microcomputers which can use product; System hardware requirements; and System software requirements: The SBCS General Ledger is designed to run on an APPLE II or APPLE II Plus with 48K of RAM, Applesoft in ROM, 2 Disk II drives on the same controller card, and a printer with either parallel or serial interface card. The manufacturer does not specify whether it will work with APPLE's Language System. APPLE's DOS is also needed (the version number is not specified).

4. Product features: This product is a conversion of the popular Osborne Associates General Ledger C-Basic package. It allows you to set up and maintain a computerized General Ledger (G/L) on the APPLE. Included are programs (1) to configure the software to the specific hardware you are using; (2) to set up and maintain a customized Chart of Accounts; (3) to enter postings to the G/L (either directly or through cash journals); and (4) to generate several reports, cash journals (for disbursements and receipts) and two of the four customary financial statements. (The Balance Sheet and Income Statement can be generated; but the Statement of Changes in Financial Position and Statement of Retained Earnings are not generated.) The G/L system allows departmentalizing of reports (useful if your business has several locations or cost centers). There is a wide latitude in formating the financial statements.

5-7. Product performance; Product quality; and Product limitations: The SBCS General Ledger performs well, though its usefulness may be limited by several factors discussed below. The system is well designed (again with certain limitations). Error trapping is excellent. I was not able to "crash" the system, though the documentation specified several conditions where the system may hang. Recovery from these situations is effected by re-booting the system. The current session's data will be lost, but previous data will not be. In converting the Osborne Associates' package, SBCS has speeded up execution and provided for different types of printers. Two separate program disks are provided; one for a Centronics-type interface (parallel) and a second disk for use with serial interfaces. If the printer being used does not have "top of form" capability (such as the Centronics 779), this function is emulated in software. A third disk, containing the sample Chart of Accounts described in the Osborne documentation, is also included. It may be used for practice on the system or may be modified for your particular business, thereby saving you the trouble of having to enter several hundred accounts.

Another addition is provision for two levels of password security. This is a nice touch when you have clerical staff operating the system—staff members cannot obtain a printout of the financial statements without knowing the second-level password.

This product does have several limitations. These result from limitations of the original Osborne software and Applesoft language, not from the conversion done by SBCS. One of the major limitations of this (and almost *all* software on the APPLE II) is that Applesoft limits you to nine digits ($9,999,999.99). While this may *not* be a problem in your business (after all, a ten-million-dollar-a-year business stretches the definition of "small business"), many businesses maintain "memo" accounts in their General Ledger. These memo accounts usually contain some sort of statistic such as units produced, units sold, etc.

The nine-digit limit may also be a problem, it you are considering using this hardware/software configuration for service bureau work or, as I do, in an accounting or bookkeeping practice. In these cases, it is quite possible that you may have clients who will have 10 million or more in any one account (such as a memo account or sales). SBCS states in both its documentation and adveritising, that it is willing and able to tailor the software for special needs. Perhaps SBCS would be willing to patch into its programs one of the existing double-precision routines available for the APPLE II, or you might wait until SBCS brings out a conversion for the APPLE III (APPLE "Business Basic" on the III has 16-digit precision).

The second major limitation of this package is the reports. While there are a multitude of them, there is no actual General Ledger produced. The closest thing to a General Ledger is the report called "G/L Update" which contains most of the information common to computerized G/L systems, but in a format that a person who is used to more conventional manuals or computer-generated G/L's might have difficulty using. This may or may not be a problem, depending on who will be using the reports. My suggestion is to purchase the Osborne book (*General Ledger C-Basic* version) before buying the software. (You will have to purchase the book anyway, it you do decide to buy the software, as it makes up the bulk of the documentation.) Sit down with your bookkeeper or accountant and see if they can live with the format of the reports.

My last major criticism of this software is that it is extremely easy to enter an unbalanced entry (credits do not equal debits) when using direct-posting entry. Most G/L software makes it very difficult to do this.

This is not a problem when entering transactions through the cash journals, as this type of entry automatically produces the correct off-setting entry. Direct posting would be used to make adjusting entries, and it is extremely easy to make a mistake here. The potential user should be cautioned to double-check each entry when using this mode. An unbalanced entry will result in the G/L being out of balance, necessitating an additional correcting entry.

8. Product documentation: Product documentation consists of two books — *The Osborne General Ledger in C-Basic,* which is not supplied with the software, and an additional 32-page manual supplied by SBCS, detailing enhancements to and differences from the C-Basic version. These two manuals comprise over 200 pages of documentation. Unfortunately, most of it is aimed at the programmer, not the user. A user with very little experience in computers and accounting (such as the average small business owner) would have a great deal of difficulty getting this package up and working. A small user's manual (15-20 pages), detailing step-by-step operations, and indexed to be a "computer-side" reference would be a welcome addition. SBCS *does* state that it expects purchasers to have *some* background in computers and accounting. And while I feel that a more user-oriented manual would be nice, the documentation supplied and available is usable (even if inconvenient); and it is much to be preferred over the flimsy or nonexistant documentation I have seen accompanying some other software.

9. Special user requirements: Purchasers of this software will find that a background in *both* computers and accounting (bookkeeping) will be useful. The better your background, the easier it will be to install and use this package. A user with *absolutely* no background in either field will probably have some difficulty getting the package up and running. A user falling somewhere in-between the two extremes may have a little difficulty at first, but should eventually get the system running. The error-trapping routines may cause some frustration, but will prevent the user from most disasters.

10. Price/feature/quality/evaluation:—This software package will not be suitable for everyone—no packaged software is. For those whose needs will be adequately served, this software at $200.00 ($180.00 for the SBCS package + $20.00 for the Osborne book) represents an excellent value, and is one of the less expensive G/L packages available for the APPLE II.

11. Additional comments: There are several excellent General Ledger packages currently on the market (BPI Systems, Apple Controller, Micro-source Ledger Plus among others). Each of these, including the SBCS conversion, has its good points and limitations. The purchaser of a software package owes it to himself *and* the producer of the software to determine whether any particular package will meet the user's needs. SBCS warranties its software against errors for one year. It also offers a 30-day, full-purchase refund, if the user finds that the software is not as documented. These are excellent warranties, but SBCS cannot guarantee that its software is exactly what *your* business needs.

Before buying *any* software sit down and determine, as precisely as possible, what you are looking for. What do I *need* this software to do? What do I *want* it to do? Does this software meet my needs and wants? Try and bring the people who will be using the software (your accountant, bookkeeper, data entry clerk) in on the decision—or at least ask for their opinion. Remember—the more you know about all the factors, the better decision you will be able to make.

Editors note— The manufacturer comments on the review as follows: SBCS General Ledger 2.1 (released August 1, 1980) eliminates some of the limitations mentioned in the review. Version 2.1 will support all APPLE printer interface cards and any printer with over 110 columns. If "top of form" is not available, it will be emulated. Version 2.1 runs under 3.2.1 DOS and may be used with the Language System. We will be offering a version for 3.3 DOS as well as the APPLE III Business Basic.

There is, however, one discrepancy in the review. Because of extensive error checking, the user cannot enter any data which will later cause the system to "crash." We are also performing error checking on the hardware, as it is not infallable. The *only* time any data will be lost is in case of a power failure or accidentally pressing "Reset" during a posting session. Then *only* the current posting will be lost. The previous postings made during the current session will *not* be affected. The condition referred to in the review will occur only if a hardware malfuction (such as a disk or printer) is detected which would result in erroneous data being generated. Recovery is as simple as reverting to the backup diskette (after correcting the malfuction).

I agree with the reviewer in that the user should first establish his or her needs. Since the Osborne manuals are readily available, one can easily see if the Osborne methodology will fulfill those needs. If not, then a major disappointment can be avoided. If their needs are *almost* met, the necessary modifications can be discussed in detail.

12. Reviewer: Ted Needleman, 67 West Burda Place, Spring Valley, NY 10977

Microprocessors in Medicine: the 6502
Part 1

by Jerry W. Froelich, M.D.

The column this month and the next, written together with Jack W. Smith, M.D., will inform readers on various uses of computers in medical education and will provide an example of how the 6502 microprocessor is able to perform tasks in medical education nearly as well as large computer systems. (Dr. Smith is a Clinical Fellow in Pathology, Instructor in Allied Health, and Ph.D. candidate in Computer Science at Ohio State University, Columbus, Ohio.)

Computers in medical education can be divided into three major categories: computer-assisted evaluation (CAE), computer-aided instruction (CAI), and simulations. These categories include testing, statistical analysis of test results, study prescriptions, tutoring, diagnosis and treatment guidance, simulation of processes, and simulation of patient-physician encounters. These serve as an extension of the classroom and not as replacement of the teacher. With the application of small, inexpensive microprocessors, such as the 6502, physicians can now acquire continuing education credits (proof of furthering their medical education to stay current with medical practice) by reviewing programs on their own computers.

The use of computers in medical education thus ranges from simple display of information to a sophisticated interaction with the physician. The discussion presented here covers only a part of that range. This month we will cover the theoretical aspects of "Computers in Medical Education" and next month we will cite examples.

CAE

CAE uses computers to handle administrative chores. The computer can administer examinations and score them immediately or grade examinations taken at a previous time. It can then make a statistical analysis of a student's performance and offer study prescriptions (references to appropriate material) to aid the student in compensating for deficiencies. Group performance can also be compiled. The interactive capability of the computer is not, however, fully realized in computer-aided evaluation.

CAI

Generally speaking, in computer-aided instruction, the computer acts as a tutor, privately coaching students and helping them acquire information in a particular subject. The computer disseminates information and tests a student's comprehension and recall. The computer can also teach and test a student on how to interpret information. For example, a CAI program could introduce a student to the physiological, biochemical, and genetic organization of bacteria, viruses, and parasites. After the student has been coached and tested on the basic concepts, the computer could present the student with a number of organisms to classify. Problem areas would prompt remedial instruction, until the student reached a previously established level of learning.

There are several advantages to presenting material in this way: (1) Faculty members are not responsible for disseminating repetitive information and are free to pursue creative endeavors. (2) CAI can be used to supplement traditional educational techniques (lecture and laboratory work), which may suffer because of budget cuts. (3) New knowledge can be incorporated more easily in the computer data base than in reference books, thus decreasing the time lapse between availability of facts and their transmission to students and physicians. (4) Students can bypass familiar material. This is especially important in medical education where students vary widely in educational backgrounds. (5) CAI is efficient, in that a student can master a subject in less time than is usually necessary with traditional methods. (6) Instruction is individual, based on the specific abilities of the student. His actions produce almost instant, positive feedback or correctional instruction. (7) Multimedia presentation is easily incorporated in this technique. Current projects in CAI involve the use of high-resolution graphic screens and computer-controlled slide projectors, as additional instructional tools.

Simulations

Educational simulations are of two varieties: simulations of biological processes (physiological, biochemical, etc.) and simulations of patient-physician encounters. Process simulation displays a model of "real-world" events, when the actual event is costly, unmanageable, or dangerous to duplicate. More importantly, a precise model of an event need not exist to simulate the event adequately for educational purposes. Simulations are a convenient way for the student to assimilate information acquired in the classroom.

An example of process simulation would be a computer program that simulates the growth of a cell system. From the computer terminal, the student can manipulate certain variables, such as death rate, mutation rate, and growth rate. The impact of a particular manipulation, in conjunction with other variables of the system, can then be instantly displayed on a computer terminal.

A computer program to simulate the patient-physician encounter can do the following: (1) present a summary of the patient's medical case or accept a case from the student; (2) allow the student to acquire information about the patient through a dialogue with the computer (this interaction would include information about the patient's history, laboratory findings, and physical exam findings); (3) display information on the availability, time, and cost of procedures needed for the patient; (4) ask the student for a preliminary diagnosis and treatment strategy or receive diagnostic and treatment advice; (5) explore the effects of such treatment along with the accuracy of the diagnosis; (6) compare the student's response to the responses of experts.

The patient-physician simulation has several advantages. The student is exposed to the problem-solving nature of clinical medicine. The simulation is without risk to patients: the student is given the opportunity vicariously to participate in patient management where clinical judgement is required. An additional merit is that management can be studied by design, rather than by the availability of patients with particular diseases.

The next column will describe several current systems used in medical education and a specific APPLE application, "APPLE-ED".

Address communications: c/o Massachusetts General Hospital, Boston, MA 02114

Ohio Scientific Users: Stop Those S┘ ERRORS

Correct the BASIC error message output, put out messages of your own, and more.

E.D.Morris Jr.
3200 Washington
Midland, MI 48640

Tim Finkbeiner
3710 Fuller
Midland, MI 48640

The original Ohio Scientific video board could display only 64 different characters: upper case letters, numbers and punctuation. The current model video board displays 256 different characters: all of the original characters plus lower case and graphics. This created an unwelcome bonus for machines which use BASIC in ROM. The error messages now appear in graphics characters rather than in letters. For example, whenever a syntax error is made, the user sees

?S┘ERROR IN LINE 10

The Ohio Scientific graphics manual explains that the correct message is

?SN ERROR IN LINE 10

I make enough syntax errors that I no longer have to look this one up. However, it becomes a real nuisance to refer to the manual for T┐ or C┘ errors. The second letter of all the error codes appears as a graphics symbol.

This article describes a patch for Ohio Scientific BASIC in ROM to convert the graphics characters in error messages back into readable letters. Three other short patches are also included that allow your BASIC to be customized in a unique way. The same technique for adding the patch to ROM BASIC is used in each program. The four programs are written in BASIC for the 540 video board. REM statements indicate changes to be made for the Superboard. The BASIC programs read data and create a machine language patch. A disassmbly of each patch is also shown. Once the BASIC program is run, it can be NEWed and the machine program will remain untouched. If the computer is cold-started, the POKEs to locations 4 and 5 must be redone. All of the patches start at hex location $0240.

PROGRAM 1

```
10 REM OK REPLACEMENT
20 DATA 169,80,160,2,76,195,168
30 FOR X=576 TO 582
40 READ Q
50 POKE X,Q : NEXT
60 INPUT"NEW MESSAGE ";A$
70 B$=CHR$(10)+CHR$(13)
80 A$=B$+A$+B$+CHR$(0)
90 A=592
100 FOR X=1 TO LEN(A$)
110 POKE A,ASC(MID$(A$,X,1))
120 A=A+1
130 NEXT
140 POKE 4,64:POKE 5,2
```

DISASSEMBLY FOR PROGRAM 1

```
0240 A950    LDA #$50
0242 A002    LDY #$02
0244 4CC3A8  JMP $A8C3
```

Before an error message can be corrected, a way must be found to break into BASIC just when the message is being printed. This is difficult since BASIC is mostly in ROM memory. There is a sneaky way of doing this, as described in the remainder of the article. Note carefully the format of error messages

?S┘ ERROR
OK

The "OK" prompt always follows the error message. To output the "OK" prompt, the BASIC interpreter jumps to $0003. At that address you will find the machine code 4C C3 A8 which means JMP $003. At that address is found the machine code 4C C3 A8 which means JMP $A8C3. According to an article in MICRO, November 1979, (18:9), $A8C3 is a subroutine to print a message. The address of the message to be output is in the A (ADL) and Y (ADH) registers. Since the locations $0003, 0004, and 0005 are in RAM, these locations can be changed to divert the computer to our own subroutine instead.

Before attempting the error correction program, let's try a simpler problem first to demonstrate the technique. Suppose we don't like the "OK" prompt. If the computer can be intercepted on its way to the message routine, the values in the A and Y registers can be changed to point to a new message of our choosing. The first BASIC program does exactly that. (If you want to convert your Ohio Scientific machine to a PET, run the BASIC program and INPUT "READY" as the new message.) Your new prompt plus appropriate line feeds and carriage returns are stored in $0250. BASIC's pointers are changed to aim at the new message. Instead of "OK" your computer will respond with "READY" or "I'M WAITING" or whatever you choose.

```
PROGRAM 2

10 REM ERROR MESSAGE CORRECTION
20 DATA 72,173
30 DATA 64,215 :REM SUPERBOARD 101,211
40 DATA 201,63,208,8,173
50 DATA 66,215 :REM SUPERBOARD 103,211
60 DATA 41,127,141
70 DATA 66,215 :REM SUPERBOARD 103,211
80 DATA 104,76,195,168,0,0
90 FOR X=576 TO 597
100 READ Q
110 POKE X,Q
120 NEXT
130 POKE 4,64 :POKE 5,2
```

```
DISASSEMBLY FOR PROGRAM 2

0240 48       PHA
0241 AD40D7   LDA $D740
0244 C93F     CMP #$3F
0246 D008     BNE $0250
0248 AD42D7   LDA $D742
024B 297F     AND #$7F
024D 8D42D7   STA $D742
0250 68       PLA
0251 4CC3A8   JMP $A8C3
```

We now have a method of detecting the "OK" prompt, but "OK" appears many times, other than after an error message. Notice that "?" appears on the line above the "OK" whenever an error is printed. After every prompt message, the computer examines the space directly above the "O" in "OK". Whenever a "?" is found, the defective character in the error message appears on the screen two spaces to the right. This graphics character can be changed into the correct letter by resetting the high order bit. Program 2 will detect when an error message appears on the screen and reset this bit to the correct character. Note the three lines which must be changed if you are using a Superboard. The disassembly is for the 540 version. If you make an error while in the SAVE mode, you will see in slow motion that the incorrect character first appears and then is corrected. With this patch in your BASIC you are now free to make all sorts of errors without fear of those funny looking graphics characters appearing. Normal graphics will not be affected.

The same method used to detect an error message can be used to sense a user input. If you enter "ABC" the computer will display

> ABC
> (blank line)
> ?SN ERROR
> OK

The user input appears 3 lines directly above"OK". The computer can check this line against a keyword. This scheme can be used to add commands to BASIC. For example, program 3 is a machine language screen clear. Once the BASIC program has been loaded and run, typing a "!" and carriage return will trigger the screen clear program. Line 50 of the BASIC program is the ASCII value of the trigger character. This can be changed to whatever you wish. Changing line 50 to "DATA 35" will allow a "#" to clear the screen.

Program 4 uses a multiple letter keyword which gets stored at $0260. A message of your choosing is stored at $0280.When you load and run the BASIC program, you must enter a "KEYWORD" and a "MESSAGE". For example you might enter "KILOBAUD" and "I LIKE MICRO BETTER". Whenever the "OK" prompt appears, the computer will search for a match to your keyword. If a match is found, your message will be output to the screen. Responding with a message is not particularly useful, except to amaze your friends. However, once you understand the technique of keyword detection, the machine program can be altered to do your bidding. You can even write a program which requires a secret password before it will run.

```
PROGRAM 3

10 REM SCREEN CLEAR
20 DATA 72,173
30 DATA 192,214 :REM SUPERBOARD 37,211
40 DATA 201
50 DATA 33 : REM ASCII TRIGGER
60 DATA 208,35,138,72,169
70 DATA 32,162,0,157,0,208,157,0
80 DATA 209,157,0,210,157,0,211,157,0
90 DATA 212,157,0,213,157,0,214
100 DATA 157,0,215,232,208,229
110 DATA 104,170,104,76,195,168
120 FOR X=576 TO 622
130 READ Q
140 POKE X,Q
150 NEXT
160 POKE 4,64:POKE 5,2
```

```
DISASSEMBLY FOR PROGRAM 3

0240 48       PHA
0241 ADC0D6   LDA $D6C0
0244 C921     CMP #$21
0246 D023     BNE $026B
0248 8A       TXA
0249 48       PHA
024A A920     LDA #$20
024C A200     LDX #$00
024E 9D00D0   STA $D000,X
0251 9D00D1   STA $D100,X
0254 9D00D2   STA $D200,X
0257 9D00D3   STA $D300,X
025A 9D00D4   STA $D400,X
025D 9D00D5   STA $D500,X
0260 9D00D6   STA $D600,X
0263 9D00D7   STA $D700,X
0266 E8       INX
0267 D0E5     BNE $024E
0269 68       PLA
026A AA       TAX
026B 68       PLA
026C 4CC3A8   JMP $A8C3
```

```
PROGRAM 4

10 REM INSERT MESSAGE ON CUE
20 DATA 72,152,72,172,63,2,185,96,2,217
30 DATA 192,214 :REM SUPERBOARD 37,211
```

```
40 DATA 208,12,136,208,245,104,104
50 DATA 169,128,160,2,76,195,168
60 DATA 104,168,104,76,195,168
70 FOR X=576 TO 607
80 READ Q
90 POKE X,Q
100 NEXT
110 INPUT"KEYWORD ";A$
120 A=608
130 POKE 575,LEN(A$)-1
140 FOR X=1 TO LEN(A$)
150 POKE A,ASC(MID$(A$,X,1))
160 A=A+1:NEXT
170 INPUT"MESSAGE ";A$
180 B$=CHR$(10)+CHR$(13)
190 A$=B$+A$+B$+CHR$(0)
200 A=640
210 FOR X=1 TO LEN (A$)
220 POKE A,ASC(MID$(A$,X,1))
230 A=A+1:NEXT
240 POKE 4,64:POKE 5,2
```

DISASSEMBLY FOR PROGRAM 4

```
0240 48     PHA
0241 98     TYA
0242 48     PHA
0243 AC3F02 LDY $023F
0246 B96002 LDA $0260,Y
0249 D9C0D6 CMP $D6C0,Y
024C D00C   BNE $025A
024E 88     DEY
024F D0F5   BNE $0246
0251 68     PLA
0252 68     PLA
0253 A980   LDA #$80
0255 A002   LDY #$02
0257 4CC3A8 JMP $A8C3
025A 68     PLA
025B A8     TAY
025C 68     PLA
025D 4CC3A8 JMP $A8C3
```

μ

OUR UNIQUE MARKETING PHILOSOPHY

MORE for *LESS*

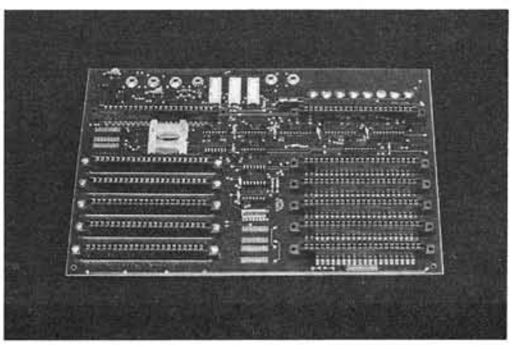
WE PROVIDE PRODUCTS FOR COST-EFFECTIVE SOLUTIONS

A fully featured mother board with a standardized bus, full buffering, address decoding for adding up to 5 expansion boards to AIM, SYM, KIM (ASK) microcomputers. MOTHER PLUS II is also a microcomputer support board providing connections for power, TTY, audio cassettes, and cassette control relays.

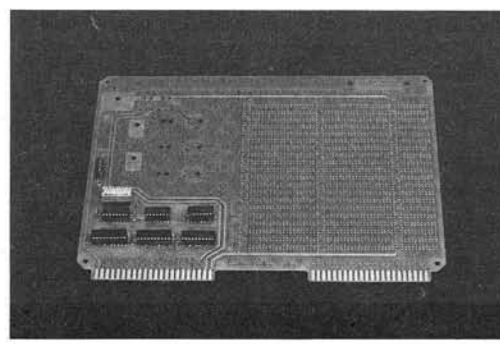
—MOTHER PLUS II—

FEATURES	BENEFITS
All address lines, data lines and control lines are buffered. Address control manager resolves host/peripheral address contention.	Address decoding and full buffering provides the necessary interface to easily add boards to increase microcomputer capabilities. No microcomputer hardware changes are required when boards are added via MOTHER PLUS II.
System I/O is supported with connectors for TTY and audio cassettes; relays to control two cassette recorders; Port A and Port B socket; LED audio input monitor, and more.	Simple switch settings prevent bus contention between the microcomputer and expansion boards.
Standard 44 pin edge connectors provide interconnection for 5 expansion boards.	The standard bus eliminates cables and permits more compact and efficient packaging.
Large terminal strip provides GND, +5V, +12V, 1 user defined voltage, and TTY I/O.	The terminal strip simplifies connections and supports special user requirements.

DEALER AND OEM QUANTITY DISCOUNTS AVAILABLE



MOTHER PLUS II: $100.



PROTO PLUS II: Unassembled-$45, Assembled-$60.

This expansion board product provides for the efficient design, development, checkout, and interface of digital design logic to AIM, SYM, KIM (ASK) microcomputers.

—PROTO PLUS II—

FEATURES	BENEFITS
Full Size board with plated through holes and gold-plated fingers. Triple pad geometry permits solder connectors as well as wire wrap. Universal .1 grid pattern with GND and +5V conveniently bussed throughout the board. Additional patterns to permit quick insertion of a wide range of discrete components, voltage regulators, 25 pin "D" subminiature plug (RS 232), common transistors, switches, etcetera.	Product development times are improved. The PROTO PLUS II board is designed to facilitate construction and assembly.
Standard interface has circuits for address buffers, bidirectional tri-state buffers, 8K bank address decoding, and control signal buffers.	Engineers can now concentrate their efforts on new design work as standard interface circuitry and components are available as standard options. Time previously lost due to long and sometimes costly order and delivery cycles is avoided.

All prices shown are US and Canadian only, and are exclusive of shipping charges and applicable taxes. Other improved products now available include: DRAM PLUS, VIDEO PLUS II, POWER A PLUS II, and AIM PLUS II. For more information, contact:

THE COMPUTERIST®
34 Chelmsford St., Chelmsford, MA 01824
617/256-3649

In this month's issue of Ohio Scientific's Small Systems Journal, we are introducing a new word processing software system—WP-3. The description, though brief in comparison to the magnitude of the system, will hopefully convey some of WP-3's tremendous word processing power.

Two new Ohio Scientific game software releases—ZULU 9 and OSI INVADERS are also described in this issue.

We are pleased to include in this issue another contributed software feature—PINBALL 2001. Our thanks to Mr. Robert Wiebe for this contribution.

The final article this month is a piece originally scheduled for the October issue of MICRO. It is a BASIC routine for OS-65D V3.2 to increase file access efficiency by up to a factor of 20.

As always, comment on article content is welcome.
Ohio Scientific, Inc.
1333 South Chillicothe Road
Aurora, Ohio 44202

Introduction to WP-3

WP-3 is Ohio Scientific's latest word processing software system. Before describing some of WP-3's specific features, let's briefly review a few general word processing concepts.

Word processing is the automated manipulation of text. This includes initial entry of text into a word processing system, editing of previously entered text and formatted printing of text. The text itself can be a form letter, a technical manual or the chapters of a book. Or it could be any other textual material that you want to print without errors, or you will be printing a number of times with minor revisions from one printing to another.

There are three basic steps involved in using a word processing system.

1. *Entry* of new text.
2. *Editing* or correcting previously entered text.
3. *Output* of previously entered text with formatting such as margin justification and page numbering.

The entry of new text into a word processing system is roughly equivalent to typing a draft of the material. Then the new text is printed for review, proofread and edited. The automatic features of the Word Processing system provide for easily making changes and automatically compensate for these changes at each printing. For example, if you insert a new sentence or paragraph, all text after the insertion is moved down and page boundaries are readjusted appropriately. Since most word processing printers print 500 or more words per minute, each printout is produced quickly and also takes little operator assistance.

Another concept implied in a word processing software system is the ability to permanently store entered text in a machine readable form. Under WP-3 text may either be stored on floppy diskettes or on a hard disk (CD-23, CD-74, etc.).

Using WP-3 the actual storage and retrieval of text data is done via named files. This means that blocks of text may be conveniently referred to by common names which have a connection to their content. Some examples could be "CHAPT1", "CHAPT2", "AFORM", "LETTER", "RESUME", etc.

Editing Features of WP-3

WP-3 has several features which greatly simplify entry and editing of text. For example, upon entry of text information, all typing may be done without concern for line length. The word processor automatically inserts all proper line terminations for easy readability on the CRT terminal.

The easiest way to demonstrate the fundamental features of WP-3 is by describing a simple session with the software.

Your first step is to initialize the text workspace. This is done with the "I" command followed by a "YES" response to "INIZ?". (This two-step procedure helps protect against unintentional initialization.)

After initialization, you type "NEW" and enter text by merely starting to type:

> It was a dark and stormy night. The wind whipped mercilessly at the sails and the howling of the wolves on the tundra touched him to the marrow.

Upon exiting the text entry mode, you may return to the top of the text file with the AGAIN command. The text may then be reviewed simply by stepping through it by typing carriage returns (or down-arrows). As each line appears on the terminal, the cursor is positioned at the beginning of the line. At this time you may either edit the line or step onto the next line.

After reviewing the text, you will probably notice that it doesn't make much sense. "Howling wolves on the tundra" while at sea appears to be ridiculous. Either the "sails" or the "wolves on the tundra" have to go.

You have several options of how to change your text. The first might be simply to remove the phrase "of the wolves on the tundra" from the body of the text. This is accomplished by inserting "markers" into the text at the beginning and at the end of the offending phrase. These markers appear in the text file as blinking vertical lines. The command DELETE will remove all characters between the markers.

Another option is to enter the line in question, delete characters and insert new characters into the line. This is done by stepping to the line, "tabbing" to the character and then removing it. The word "sails" could be removed, for example, and the word "igloo" typed in.

As a final option, a block of text could simply be changed to other text by using the CHANGE command. You could simply type CHANGE "sails", "flimsy cabin walls". This would replace the word "sails" with the phrase "flimsy cabin walls".

There are several other editing commands that are extremely useful. Unfortunately, they don't lend themselves very well to our simple example, so a description will have to suffice.

	LIST	ZIGZAG
LINE PRINTER	XTRA OFF ADD SPACES BETWEEN WORDS TO RIGHT JUSTIFY	XTRA OFF RAGGED EDGE TO LIMITED RIGHT MARGIN
	XTRA ON NOT ALLOWED	XTRA ON NOT ALLOWED

	LIST	ZIGZAG
SERIAL WORD PROCESSING PRINTER	XTRA OFF ADD SPACES BETWEEN WORDS TO RIGHT JUSTIFY	XTRA OFF RAGGED EDGE TO LIMITED RIGHT MARGIN
	XTRA ON FINELY GRADUATED SPACING BETWEEN WORDS AND LETTERS TO RIGHT JUSTIFY	XTRA ON SAME AS XTRA OFF

	LIST	ZIGZAG
PARALLEL WORD PROCESSING PRINTER	XTRA OFF FINELY GRADUATED SPACING BETWEEN WORDS AND LETTERS TO RIGHT JUSTIFY	XTRA OFF RAGGED EDGE TO LIMITED RIGHT MARGIN
	XTRA ON SAME AS XTRA OFF WITH ADDITIONAL PROPORTIONAL CHARACTER SPACING	XTRA ON SAME AS XTRA OFF WITH ADDITIONAL PROPORTIONAL CHARACTER SPACING

Figure 1:
WP-3 Output Format Table

The FIND command will find the first occurrence of specified text. All remaining occurrences may be located by re-commanding FIND with no new text specification.

The MOVE and TRANSFER commands manipulate the location of blocks of text. A block of text may be moved by first defining its start and end with markers (described previously) and then locating where the text should be moved to with the cursor. That is, the marked text will be moved such that it will immediately follow the current cursor location. The TRANSFER command works the same way, but leaves a copy of the text at its original location.

Output-Formatting Features of WP-3

After the entry of your text is complete, you will undoubtedly want some sort of permanent copy of your work. WP-3 supports three types of printed output:

Lineprinter (Centronix-type interface)
Serial Word Processing Printer
Parallel Word Processing Printer

This is, of course, in addition to the standard CRT terminal output.

There are two basic output formatting commands. They are LIST and ZIGZAG. LIST outputs lines of uniform length while ZIGZAG outputs lines with "ragged" right margins. Each of these commands require a width parameter. LIST (width) defines the maximum line length. Parameters for page numbers, output device, etc., are optional.

Another pair of commands, XTRA ON and XTRA OFF controls the proportional spacing formatting of the output when used with a serial or parallel word processing printer.

The command HYPHENATE (count) allows automatic hyphenation of words at the end of lines after (count) characters in the word. HYPHENATE may be used with either LIST or ZIGZAG.

The chart in Figure 1 describes the various outputting options versus output device.

There are also several commands which allow control of the lines per page, spacing between lines, hold output at end of page, etc.

A unique feature in the output formatting routine is the "embedded command" ACCEPT. When this command code has been "embedded" into your normal text and is found during a LIST or ZIGZAG output, the printing stops and waits for an input from the terminal. Upon completion of terminal input, whatever you have typed in is printed before continuation of the standard printed text.

A number of other commands may be "embedded" into your normal text to control WP-3's output formatting. These include command codes for indentation, tabination, pagination, skip specified lines, underline, etcetera. With all these commands, the action is taken without printing the "embedded" command code.

Of necessity, this has been a very brief description of a few of WP-3's many features. You should contact your nearest Ohio Scientific dealer for further details concerning WP-3 and the recommended system configuration to fully utilize this powerful word processing package.

OSI Invaders and Zulu 9

OSI offers nearly one-hundred programs for its personal computer line. This range from battleship to tanks; including action games (like bomber and hectic), sports simulations (like bowling and golf), card games (spaces and hearts, for instance), strategy games (try Othello or chess) and arcade-type games. This month we will highlight two of the arcade-type games: Zulu and OSI Invaders.

Zulu 9, written in assembler, is a unique rendition of the interstellar pursuit theme made popular by the movie Star Wars. You are given the controls of a powerful star ship—your objective is to destroy as many invading alien ships as possible without running out of energy. On the disk version your controls consist of two joysticks to steer, accelerate, decelerate and fire your lasers. At the start of the game you have to choose your handicap (25 for beginners, 0 for experts), vertical retrace option (this selects optimum video display for color televisions) and whether or not collisions with invading space craft are allowed.

You will begin with 100% energy at speed 10. Speeds from 1-10 deplete your fuel and from 11-20 replenish the fuel supplies. The faster you go, the harder the incoming crafts are to destroy. The screen depicts your view of space from the cockpit of your star ship. The direction controls act like the control stick of an airplane. As you fly through the stars you will find that the alien's shields protect him from all hits except to the center of his ship—your shots have to be right "on target". Another interesting feature, your speed relative to the alien vessel, will determine whether he's getting closer or further away.

Zulu 9 is available on GD-8 with three other games for disk machines, black and white and color compatible with built-in DAC sound effects for $35.00. The cassette version, which costs $9.00, is a 4K black and white program and does not require joysticks.

OSI Invaders is a new release. Starting with three turrets, fire your laser cannon at the hoard of alien invaders as they relentlessly march across the screen coming closer and closer, constantly dropping bombs on you and your shields. This popular game (written in assembler) offers 15 levels of play from slow to very, very fast. Each time you clear the screen you will get another turret (up to nine maximum) but it gets harder because the invaders come faster each time and the fewer you can see, the faster they go!

This is a one player game that is played from the keyboard. Current score, turret count, and high score are constantly displayed. Disk versions store the high score for each level (cassettes do not). The cassette costs $19.00, runs in 8K on C1P's, C4P's and C8P's (program does not use color or sound). The disk is available for C1P's, C4P's and C8P's for $29.00.

Pinball 2001

Many users of Ohio Scientific's personal computers submit programs for our consideration. This one was authored by Mr. Robert Wiebe of Canada. The instructions are self-contained and complete. This is easily converted to BASIC-in-ROM machines by modifying lines 160, 161 and 2455. The POKEs contained in those lines may be new to some readers: POKE 9770,0 disables the scroll and POKE 9770,64 restores it.

Some interesting visual effects can be produced by experimenting with these POKEs. Try the following for starters:

```
10 FOR SC = 1 TO 30:?:NEXT:A = 9770
20 FORI = 0 TO 255:POKE A,I:?"*":NEXT
30 POKE A,64
```

Remember to POKE 9770,64 when you are done experimenting with various STEP rates in line 20.

```
10 REM PINBALL 2001
20 INPUT"Do you want instructions (Y/N)";A$
25 IF LEFT$(A$,1)="Y"THEN2500
30 FORX=1TO30:PRINT:NEXT
40 FORX=0TO43:POKEX+53514,155:POKEX+54986,154:NEXT
60 FORX=54018TO54466STEP64:POKEX,156:POKEX+59,157:NEXT
70 FORX=53558TO54013STEP65:POKEX,169:POKEX+1,96:NEXT
75 FORX=53513TO53954STEP63:POKEX,170:POKEX-1,96:NEXT
80 FORX=54589TO55030STEP63:POKEX,170:POKEX+1,96:NEXT
85 FORX=54648TO54837STEP63:POKEX,189:POKEX+64,96:NEXT
90 FORX=54530TO54985STEP65:POKEX,169:POKEX-1,96:NEXT
95 FORX=53588TO53716STEP64:POKEX,233:POKEX+23,233:NEXT
100 FORX=53721TO53849STEP64:POKEX,143:POKEX+1,136:POKEX+6,143
105 POKEX+7,136:POKEX+12,143:POKEX+13,136:NEXT
110 FORX=54599TO54804STEP65:POKEX,190:POKEX+1,96:NEXT
115 FORX=54795TO54797:POKEX,128:POKEX+39,128:NEXT
120 FORX=54674TO54678:READA:POKEX,A:POKEX+23,A:NEXT
121 Q=125
125 A=54110:B=3:FORX=1TO3:FORY=0TOB:POKEA+Y,Q:NEXTY:A=A+63
130 B=B+2:NEXTX:A=A-64
135 FORX=1TO3:A=A+65:B=B-2:FORY=0TOB:POKEA+Y,Q:NEXTY,X
140 A$=CHR$(13):PRINTSPC(63)A$;:POKE55167,32
145 A=53961:FORX=0TO7:POKEX+A,4:POKEX+A+38,4:NEXT
150 FORX=1TO10:READA:READB:POKEA+54154,B:POKEA+54196,B
155 NEXT
160 POKE2073,96:A=57088
161 POKE9770,0
165 B=6
200 B=B-1:PRINTTAB(47)"BALLS:"B:IFB=0THEN2400
201 PRINTTAB(22)"Hit <SPACE> for ball"
202 PRINTTAB(9)"SCORE: "S
205 F=0:FORX=54992TO55023:POKEX,154:NEXT
210 FORX=1TO50:IFPEEK(57088)=16THEN220
211 POKEA,2:NEXT
215 PRINTSPC(60):PRINT
216 FORX=1TO200:NEXTX:GOTO201
220 PRINTSPC(60):PRINT
230 POKE53611,32:C=53620:D=-1
240 FORX=1TO31:C=C+D:POKEC,226:POKEC+1,32:NEXT
250 POKE53611,233:FORX=1TOINT(RND(C)*19+1)
260 C=C+1:POKEC,226:POKEC-1,32:NEXT
270 D=65:POKEA,1
300 P=PEEK(A):IFP<2ORP>7THEN400
301 IFF=1THEN400
302 FL=184:F=1
305 IFP=7THEN375
```

```
310 IFP=3THEN325
315 FORFF=55007TO54992STEP-1: POKEFF, FL: GOSUB400: NEXT
320 FORFF=54992TO55007: POKEFF, 154: GOSUB400: NEXT: F=0: GOTO300
325 FORFF=55008TO55023: POKEFF, FL: GOSUB400: NEXTFF
330 FORFF=55023TO55008STEP-1: POKEFF, 154: GOSUB400: NEXT
335 F=0: GOTO300
340 IFF=1THENRETURN
360 GOTO300
375 F1=55007: F2=F1+1: FORFF=0TO15: POKEF1-FF, FL: POKEF2+FF, FL
380 GOSUB400: NEXT
390 FORFF=15TO0STEP-1: POKEF1-FF, 154: POKEF2+FF, 154: GOSUB400: NEXT
395 F=0: GOTO300
400 C=C+D: IFPEEK(C)<>32THENP=PEEK(C): C=C-D: GOTO500
401 X=RND(2): IFX>.5THENX=64
402 IFX<.5THENX=-64
420 POKEC, 226: POKEC-D, 32
421 DC=DC+1: IFDC<>10THEN440
422 DC=0
425 IFPEEK(C+X)<>32THEN440
430 C=C+X: POKEC, 226: POKEC-X, 32
440 IFF=1THENRETURN
460 GOTO300
500 IFP=154THENPOKEC, 32: GOTO200
501 IFP=128THENC=C-128: POKEC, 226: POKEC+128, 32: D=-D: GOTO400
505 IFP<>FLTHEN540
510 X=RND(A): IFX<.5THENX=1
515 IFX>.5THENX=-1
520 C=C+X: POKEC, 226: POKEC-X, 32
530 IFD=63THEND=-65: GOTO400
535 IFD=65THEND=-63: GOTO400
540 S=S+P: PRINTTAB(9)"SCORE: "S: IFP<136ORP=155THEN545
541 GOTO600
545 IFD=65THEND=-63: GOTO400
550 IFD=-65THEND=63: GOTO400
555 IFD=-63THEND=65: GOTO400
560 IFD=63THEND=-65: GOTO400
600 IFD=65THEND=63: GOTO400
605 IFD=-65THEND=-63: GOTO400
610 IFD=63THEND=65: GOTO400
615 IFD=-63THEND=-65: GOTO400
2000 DATA40, 42, 42, 42, 41
2010 DATA0, 221, 1, 222, 64, 140, 65, 139, 128, 140, 129, 139, 192, 140, 193
2020 DATA139, 256, 220, 257, 223
2400 B$="YOUR SCORE: "+STR$(S): BB=32-INT(LEN(B$)/2)
2401 PRINTSPC(60)
2402 PRINT
2410 PRINTTAB(BB)B$
2415 PRINTTAB(19)"HIT <SPACE> TO PLAY AGAIN"
2420 FORX=1TO1500: NEXT: PRINTSPC(60): FORX=1TO500: NEXT
2421 PRINT
2435 PRINTTAB(18)"HIT <RETURN> TO END THE GAME"
2440 FORX=1TO1500: NEXT: PRINTSPC(60): FORX=1TO500: NEXT
2441 PRINT
2450 POKEA, 255: BB=PEEK(A): IFBB=17THENCLEAR: RESTORE: GOTO40
2455 IFBB=9THENPOKE9770, 64: RUN"BEXEC*
2460 GOTO2415
2500 FORX=1TO11: PRINT: NEXT: PRINTTAB(28)"PINBALL  2001"
2505 PRINT: PRINT: PRINT
2510 PRINT"It is a simple game of Pinball in which you control"
2520 PRINT"the flippers and the computer controls the ball."
2530 PRINT
2540 PRINT"To control the left hand paddle use the left <SHIFT>"
2550 PRINT"To control the right paddle use the right <SHIFT>"
2560 PRINT"To use both paddles at the same time use both <SHIFTS>"
2570 PRINT"at the same time (hold them both down)."
2580 PRINT: PRINT: PRINT
2590 PRINT"Everything else you need to know is written into the"
2600 PRINT"program, so just follow it's instructions and you'll"
2610 PRINT"be o.k.": PRINT: PRINT
2620 FORX=1TO5: PRINT: NEXT
2630 PRINT"PRESS <Y> FOLLOWED BY <RETURN>": INPUTA$
2640 GOTO35
```

OS-65D V3.0 'DISK GET' Subroutine

One of the many extensions to BASIC in OS-65D is the DISK GET command which is used in conjunction with random access data files. The effect of the command is this: one track of data is loaded into RAM and the memory I/O pointers are set to the beginning of the record which was requested. Unfortunately, if the record you request is already in RAM, the track will still be reread when the DISK GET is encountered. Hence, sequential or nearly-sequential access of random files can become very time consuming.

This subroutine allows for sequential access to random files at a speed comparable to strictly sequential files. The PEEKs and POKEs used, as well as the DISK GET command itself, are listed in the OS-65D User's Guide, page 8. The operation of the subroutine is as follows:

1. Open the file as usual—DISK OPEN,6,filename.
2. Set the record size as usual. (The record size will default to 128 bytes.)
3. Set the variable RN to the number of the record you wish to access.
4. GOSUB10000—Transfer control to the DISK GET subroutine.
5. Repeat 3-4 as desired.
6. Close the file as usual—DISK CLOSE,6.

The subroutine differs from the actual DISK GET command in the following respects:

1. No redundant disk reads are executed, that is, if records 5 and 7 are on the same track, that track will be read only once if both records are requested sequentially.
2. A DISK GET which requires another track to be read will involve a DISK PUT operation if any information currently in the buffer has been altered.

This subroutine is designed as an aid to home users of Ohio Scientific machines. Although this routine has been thoroughly tested, it is not suggested for use by the beginning computer enthusiast. It is strongly recommended that the user become familiar with standard data file techniques before moving on to this useful extension.

```
10000 DEF FNA(X)=10*INT(X/16)+X-16*INT(X/16)
10010 DEF FNB(X)=16*INT(X/10)+X-10*INT(X/10)
10020 TR=INT(RN/PEEK(12042))
10030 IF FNA(TR+FNB(PEEK(9002)))=PEEK(9004) THEN 10060
10040 IF PEEK(9005) THEN DISK PUT
10050 DISK GET, RN : RETURN
10060 RA=(RN-TR*PEEK(12042))*(2^PEEK(12076))+PEEK(8998)+PEEK(8999)*256
10070 AH=INT(RA/256) : AL=RA-AH*256
10080 POKE 9132, AL : POKE 9133, AH : POKE 9155, AL : POKE 9156, AH : RETURN
```

A Versatile Hi-Res Function Plotter for the ATARI 400 and 800

The ATARI offers many possibilities with its color graphics. The discussion and program provide a starting point for understanding and utilizing these potentials.

David P. Allen
19 Damon Road
Scituate, MA 02066

In the September 1980 issue of MICRO (28:39) I presented a program for the APPLE II which plotted an infinity of trigonometric functions (and other functions as well) in the Hi-Res mode. Not long after I developed that program I obtained the new ATARI 400 computer. I was immediately impressed by the sophisticated graphics routines contained in ATARI BASIC and I decided to see how well some of my APPLE II graphics programs would translate into ATARI BASIC. The answer is ... very well, thank you!

While APPLE II has three screen modes (text, Lo-Res graphics, and Hi-Res graphics) the ATARI has *nine* screen modes and each has a greater number of permutations than does the APPLE II. This does not come completely unfettered by problems, for getting the graphic capability out of the ATARI machine is much more complicated than with the APPLE II. After telling it which of the two graphic modes you're interested in, APPLE asks only what color to plot and where to plot it. ATARI is interested in these things, and also the color of the background, the color of the border outside the graphics window, and the luminance, or brightness, value of the plot, the background, and the border. And ATARI offers you not two grades of resolution, but *four!*

For comparison, see figure 1.

Since the ATARI 400 comes with only 8K of RAM it soes not have enough available memory to support the GR.8 mode. So, my first translation from APPLE II Hi-Res graphics was to the substantially lower resolution of ATARI GR.7 mode. The conversion turned out to be quite easy and is contained in the listing. Lines 100 through 250 set the graphics parameters and, as set, will display the graph in orange (white, on black and white screens) on a black background. Change line 210 to Setcolor 2, 12, 4 and line 250 to Setcolor 4, 12, 4 and you will print the graph against a pleasant green background. Tough to do on an APPLE, easy to do on an ATARI.

I find the results of this lower resolution plot to be quite acceptable. Highly complex waveforms can get badly muddled up at times, but changing line 50 to expand the muddled area can reveal the covered up detail. For example, if a 1- to 360-degree plot is inconclusive in the 45- to 60-degree range, then substitute 45 for 1 and 60 for 360 in line 50, and run the program again. This will cause the area in question to be expanded across the entire screen.

You can have greater resolution by stepping at rates of 1 or less in line 2100.

After this first translation I acquired an ATARI 800 computer with 48K of memory so I decided to see what would happen with a GR.8 version of this program. It comes off very well and, of course, has much higher resolution to offer than APPLE's Hi-Res mode. We are limited in the GR.8 mode to only two colors, namely white and something else for the background, but I do not find this to be particularly restricting. With more points to plot it takes more time, but much greater detail can be obtained, especially with the magnification techniques described above.

APPLE II		ATARI	
Mode	**Resolution**	**Mode**	**Resolution**
GR	40 × 48	GR.3	40 × 24
HGR	280 × 192	GR.4 (or 5)	80 × 48
		GR.6 (or 7)	160 × 96
		GR.8	320 × 192

Figure 1

Here are the program listings for the function plotting program in modes GR.7 and GR.8. The GR.8 version can be used with the ATARI 400 only if it is equipped with the accessory 8K memory, which makes the 400 a 16K machine.

So try these out on your ATARI machines. Eliminate the REM statements and save vast amounts of memory. Try fooling around with For... Next loops around line 2900 and get an integrated plot with variable changes. Lots of things are possible here. Have fun!

David Allen's publications include Television System Design *for the United States Air Force. As a contributing editor to* Video Magazine, *he writes both articles and a monthly production column.*

```
1 REM FUNCTION PLOTTER PROGRAM
2 REM BY DAVID P. ALLEN
3 REM ATARI FLOATING POINT BASIC
4 REM COPYRIGHT (C) 1980.
5 REM
6 REM THIS PROGRAM PLOTS A
7 REM CURVE FOR ANY EXPRESS-
8 REM ION AS A FUNCTION OF
9 REM INCREASING ANGLE FROM
10 REM 1 TO 360 DEGREES.
11 REM CHANGE LINE 2900
12 REM TO A FUNCTION YOU WISH
13 REM TO PLOT.
14 REM
15 REM
40 REM ESTABLISH GRAPH STARTING
41 REM AND ENDING POINTS.
42 REM
43 REM
50 R1=1:R2=360
88 REM
89 REM
90 REM SET GRAPHIC PARAMATERS
91 REM
92 REM
100 GRAPHICS 7
200 COLOR 1
210 SETCOLOR 2,0,0
250 SETCOLOR 4,0,0
268 REM
269 REM
270 REM PLOT GRAPH AXIS
271 REM
272 REM
300 PLOT 1,1:DRAWTO 1,80
400 PLOT 1,40:DRAWTO 157,40
500 FOR I=0 TO 80 STEP 10
600 PLOT 1,I:DRAWTO 3,I
700 NEXT I
800 FOR I=1 TO 158 STEP 39
900 PLOT I,38:DRAWTO I,42
1000 NEXT I
1100 REM
1110 REM
1120 REM SET FLAGS FOR FIRST PLOT
1130 REM AND SCALE.
1140 REM
1150 REM
2000 F=0:G=0
2010 REM
2020 REM
2030 REM START PLOTTING
2040 REM
2050 REM
2060 REM CHANGE STEP FOR MORE
2061 REM OR LESS RESOLUTION.
2062 REM IF R1> R2 THEN STEP
2063 REM MUST BE NEGATIVE
2064 REM (PRECEDED BY A MINUS
2065 REM SIGN).
2066 REM
2067 REM
2100 FOR I=R1 TO R2 STEP 3
2110 REM
2120 REM
2130 REM NEXT THREE STEPS ESTABLISH
2140 REM HORIZONTAL SCALE.
2150 REM
2160 REM
2200 IF ABS(R1)>=ABS(R2) THEN R=ABS(R1)
2300 IF ABS(R2)>=ABS(R1) THEN R=ABS(R2)
2400 IF G=0 THEN S=158/R:G=1
2500 X=I:Y=0
2550 REM
2551 REM
2552 REM CONVERT DEGREES TO
2553 REM RADIANS.
2554 REM
2555 REM
2600 X=X*3.14159/180
2650 REM
2651 REM
2652 REM PREVENTS CRASHING WHEN
2653 REM X = 0.
2654 REM
2655 REM
2800 IF X=0 THEN X=1.0E-05
2850 REM
2851 REM
2852 REM NEXT LINE DESCRIBES
2853 REM FUNCTION TO BE PLOTTED.
2854 REM
```

```
2855 REM
2900 Y1=SIN(X)*COS(X^2)
3000 Y=Y+Y1
3100 Y=Y*20
3150 REM
3151 REM
3152 REM SCALES X
3153 REM
3154 REM
3200 X=I*S
3250 REM
3251 REM
3252 REM RELATES PLOT TO X AXIS.
3253 REM
3254 REM
3300 Y=-Y+40
3350 REM
3351 REM
3352 REM SUBROUTINE PREVENTS
3353 REM OFF-SCALE CRASHING.
3354 REM
3355 REM
3400 GOSUB 5000
3450 REM
3451 REM
3452 REM PLOTS FIRST POINT.
3453 REM
3454 REM
3500 IF F=0 THEN PLOT X,Y:F=1
3600 DRAWTO X,Y
3700 NEXT I
3750 REM
3751 REM
3752 REM DISPLAYS EQUATION OF
3753 REM PLOTTED FUNCTION BENEATH
3754 REM GRAPHIC DISPLAY.
3755 REM
3756 REM
3800 LIST 2900
3900 END
5000 IF X<0 THEN X=0
5100 IF X>158 THEN X=158
5200 IF Y<0 THEN Y=0
5300 IF Y>80 THEN Y=80
5400 RETURN

1 REM FUNCTION PLOTTER PROGRAM
2 REM BY DAVID P. ALLEN
3 REM ATARI FLOATING POINT BASIC
4 REM COPYRIGHT (C) 1980.
5 REM
6 REM THIS PROGRAM PLOTS A
7 REM CURVE FOR ANY EXPRESS-
8 REM ION AS A FUNCTION OF
9 REM INCREASING ANGLE FROM
10 REM 1 TO 360 DEGREES.
11 REM CHANGE LINE 2900
12 REM TO A FUNCTION YOU WISH
13 REM TO PLOT.
14 REM
15 REM
40 REM ESTABLISH GRAPH STARTING
41 REM AND ENDING POINTS.
42 REM
43 REM
50 R1=1:R2=360
88 REM
89 REM
90 REM SET GRAPHIC PARAMATERS
91 REM
92 REM
100 GRAPHICS 8
200 COLOR 3
250 SETCOLOR 1,1,14
251 SETCOLOR 2,0,0
252 SETCOLOR 4,0,0
268 REM
269 REM
270 REM PLOT GRAPH AXIS
271 REM
272 REM
300 PLOT 1,1:DRAWTO 1,160
400 PLOT 1,80:DRAWTO 314,80
500 FOR I=0 TO 160 STEP 20
600 PLOT 1,I:DRAWTO 6,I
700 NEXT I
800 FOR I=0 TO 316 STEP 79
900 PLOT I,76:DRAWTO I,84
1000 NEXT I
1100 REM
1110 REM
1120 REM SET FLAGS FOR FIRST PLOT
1130 REM AND SCALE.
1140 REM
1150 REM
2000 F=0:G=0
2010 REM
2020 REM
2030 REM START PLOTTING
2040 REM
2050 REM
2060 REM CHANGE STEP FOR MORE
2061 REM OR LESS RESOLUTION.
2062 REM IF R1> R2 THEN STEP
2063 REM MUST BE NEGATIVE
2064 REM (PRECEDED BY A MINUS
2065 REM SIGN).
2066 REM
```

```
2067 REM
2100 FOR I=R1 TO R2 STEP 3
2110 REM
2120 REM
2130 REM NEXT THREE STEPS ESTABLISH
2140 REM HORIZONTAL SCALE.
2150 REM
2160 REM
2200 IF ABS(R1)>=ABS(R2) THEN R=ABS(R1)
2300 IF ABS(R2)>=ABS(R1) THEN R=ABS(R2)
2400 IF G=0 THEN S=316/R:G=1
2500 X=I:Y=0
2550 REM
2551 REM
2552 REM CONVERT DEGREES TO
2553 REM RADIANS.
2554 REM
2555 REM
2600 X=X*3.14159/180
2650 REM
2651 REM
2652 REM PREVENTS CRASHING WHEN
2653 REM X = 0.
2654 REM
2655 REM
2800 IF X=0 THEN X=1.0E-05
2850 REM
2851 REM
2852 REM NEXT LINE DESCRIBES
2853 REM FUNCTION TO BE PLOTTED.
2854 REM
2855 REM
2900 Y1=SIN(X)*COS(X^2)
3000 Y=Y+Y1
3100 Y=Y*20
3150 REM
3151 REM
3152 REM SCALES X
3153 REM
3154 REM
3200 X=I*S
3250 REM
3251 REM
3252 REM RELATES PLOT TO X AXIS.
3253 REM
3254 REM
3300 Y=-Y+80
3350 REM
3351 REM
3352 REM SUBROUTINE PREVENTS
3353 REM OFF-SCALE CRASHING.
3354 REM
3355 REM
3400 GOSUB 5000
3450 REM
3451 REM
3452 REM PLOTS FIRST POINT.
3453 REM
3454 REM
3500 IF F=0 THEN PLOT X,Y:F=1
3600 DRAWTO X,Y
3700 NEXT I
3750 REM
3751 REM
3752 REM DISPLAYS EQUATION OF
3753 REM PLOTTED FUNCTION BENEATH
3754 REM GRAPHIC DISPLAY.
3755 REM
3756 REM
3800 LIST 2900
3900 END
5000 IF X<0 THEN X=0
5100 IF X>316 THEN X=316
5200 IF Y<0 THEN Y=0
5300 IF Y>160 THEN Y=160
5400 RETURN
```

Up From the Basements

by Jeff Beamsley

Though this column is being written in the heat of late summer, it will appear in late fall. For those of you who own department stores, late fall is just before that joyful time of uncontrolled consumption called Christmas. This will also be the first Christmas that the mass marketers will be involved in personal computers in a big way. In the thick of it, of course, is Ohio Scientific.

Large retailers have not had very pleasant experiences with home computers. Sears Roebuck and Co. made several tentative attempts to sell machines. Its latest liason was with Atari. At last report, Sears had pulled the Atari machines out of its stores because of the problems store personnel had selling and supporting the machines. Sears has since set up a special training program to educate its salespeople in the finer points of computer marketing. All of the retail computer stores had a chortle over that. But Sears and others did not get to be large multi-million dollar corporations by making silly mistakes. Where there is a dollar bill, there is a way.

Working under that philosophy, Ohio Scientific and Montgomery Ward & Co. have devised a solution to the problem. Their solution takes advantage of the "client store" philosophy used to justify the insurance booths, optical centers, restaurants, and specialty shops present in many department stores. These activities are not owned by the store. The operators rent the floor space for some percentage of the gross and provide the furniture and personnel. This same approach with staff and financing from local distributors and dealers will be producing Ohio Scientific computer shops in Ward's stores all over the country from now through Christmas.

Montgomery Ward is just the beginning, though. Every corporate president and his accountant read of the 650% growth enjoyed by Apple Computer Co. last year. They are all going to be eagerly watching this Christmas season, expecting to enjoy the same success. Digital Equipment Corporation (DEC) has already opened a number of retail stores around the country. Xerox Corporation is rumored to be taking the same path, as a result of a marketing agreement with Apple. Not to be outdone, Ohio Scientific is also represented among the biggies. CDC, that's spelled Control Data Corporation, is opening ten retail stores nationwide to market its PLATO systems and Ohio Scientific equipment. The CDC stores will also serve as regional repair depots for Ohio Scientific personal machines.

How do all of these fireworks affect you and me? Among other things, Ohio Scientific products will probably enjoy the biggest boost in credibility since Clark Kent discovered the phone booth. If the Montgomery Ward program is even marginally successful, there will be a very large number of new Ohio Scientific users coming into the marketplace. The average store must produce twelve to fifteen users a month to break even. Multiply that by the hundred or so stores that are scheduled to be open by the Christmas season, and you get an idea of the potential of the market.

These new users will demand services from the marketplace in the form of software, additional documentation, and support. Ohio Scientific has already contracted with Howard W. Sams & Co., Inc. to rewrite its personal computer manuals in anticipation of this demand. Ohio Scientific has created a new machine, the C4P-DF, to better bridge the gap between the personal machine and its line of business computers. The company has also repackaged the C1P, added some features, and increased the retailer's margin. The new machine is called the C1P series II. CDC conveniently falls into place as the regional service center. CDC also has a very large library of excellent software created on its PLATO system. The company is rumored to be in the process of translating large portions of that library to run on Ohio Scientific systems — just in time to meet the anticipated demand.

We are already seeing a significant increase in independent vendors producing products for Ohio Scientific personal machines. I can't vouch for the quality of all of the software, or the advisability of some of the modifications that are advertised, but the fact that they are being advertised nationally implies that the market for such things is expanding. The influx of new users due this fall, combined with the pressure for quality documentation from Montgomery Ward and the high quality software and support due from CDC, should produce a whole new class of Ohio Scientific users. We will see the Ohio Scientific user who brags about his machine, the user who is impressed by the quality of the documentation as well as the hardware, and the user who buys the machine for the large library of software available.

Whether you like it or not, this is the user who will make up the phalanx of the personal computer invasion into the home. This is also the user that will determine the direction of the marketplace. The swelling numbers of this type of user will finally compel manufacturers to behave in a responsible way.

It is not a new age, but is is certainly a new face. If the mass market is as ripe for exploitation as the projections say, that face is sure to have a smile on it.

Please send all comments to:

Jeff Beamsley
c/o The Software Federation
44 University Drive
Arlington Heights, IL 60004

John Conway's Game of Life Using Display Devices with Automatic Scrolling

Life has been presented before for specific video displays. Here are the routines necessary to run Life on any general display device.

Theodore E. Bridge
54 Williamsburg Drive
Springfield, MA 01108

This is a much improved version of a previous article on the game of Life that was published in MICRO February 1979 (9:39). You can easily adapt this program for any 6502 computer by changing jump instructions between addresses 2096—20AF. You can use any display device, even a printer, if it will automatically roll the display upwards after the bottom line is printed.

The program is very fast. A carriage return occurs as soon as there are no more characters to be printed on a line. Moreover, two lines in the pond are printed as only one line on the display. Refer to the examples to see how this works.

Furthermore, you can change parameters in the program to adjust to the size of your display. Also, you can skip one or more generations between printings.

ADDR	Parameter Name	Default Value	Description
2001	CPL	$20	Insert the number of characters per line in your display.
2005	LIS	$10	Insert the number of lines in your screen.
2009	GPB	$00	Insert the number of generations to be skipped between printings.

Martin Gardner published John Conway's game of Life in the October and November, 1970 issues of the *The Scientific American.* Our two examples were taken from his article.

We like to think of the game of Life as a computer simulation of a virus growing on a pond of DNA, using Conway's genetic rules, which are:

1. An empty cell having exactly 3 neighbors will give birth to a new cell.
2. A living cell having less than 2, or more than three neighbors will die.
3. All births and deaths occur at one time at the end of each generation; after all cells have been examined.

We kill all cells that touch the bank of the pond. This is necessary to prevent wrap-around. The pattern would be badly damaged if wrap-around growth were allowed to collide with the main organism. Because of our rather small pond, the display in our example 2 has already departed from the original pattern produced on an infinite pond.

The program occupies $298 bytes of RAM. The pond immediately follows the program. The following space is needed for the pond: 2*(CPL *(LIS + 1)).

After loading the program, start at address 2000 and depress "G". The computer will respond with "ENTER V,H?". This is your cue to start entering the verticle and horizontal coordinates for each living cell in the seed group that you want to start with. This is your way of planting the seed of the organism that you want to study.

These coordinates are displacements from an origin at the center of the screen. Positive directions are down and to the right. A coordinate may be any decimal digit less than "8", followed by a minus sign "-", if negative; or a space if positive. If you make a mistake, enter the letter "X" to erase the entry. (Any letter may be substituted for "X".)

After you have entered coordinates for all of the living cells in the arrangement you want to start with, depress slash "/", and you are off and running.

The following two examples were given in Gardner's article:

***Example 1:* the famous traffic light. It is plotted on a pond 16 × 16.**

```
KIM
2000 A9 2001
2001 20 10.
2002 85 2005
2005 10 8.
2006 86 2000
2000 A9 C

                    ';'

          0002

                     ,
                    ;;;

          0003

                    ,,,
                    ' '
                     '

          0004

                     ;
                    ','
                     '

          0005

                    ,,,
                    ;,;

0006

         '
        ' '
       '   '
        ' '
         '

0007

        ,;,
       ';,;'
         '

0008

       ,''',
       ;   ;
        '''

0009

          ,
        ',;',
      ';',';'
        ';'

0010

        ,,,
     ,       ,
     ;       ;
        ,,,

0011

          ;
          '
     '''     '''
          ;
          '

0012

        ,,,
     ,       ,
     ;       ;
        ,,,
```

***Example 2:* the R pentomino that was plotted to its death after 1103 generations at the Case Western Reserve University with a computer program by Gary Filipski and Brad Morgan, with the results sent in by Ranan B. Banerji. It has produced 6 gliders before death. Here we plot every tenth generation on our ASR 33 TTY at 110 BAUD.**

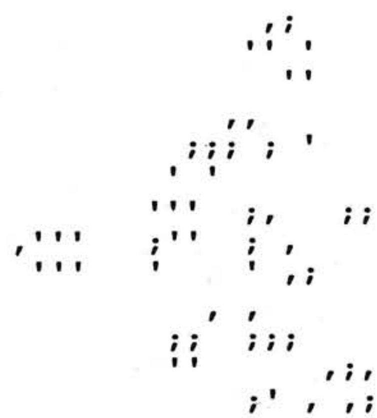
```
2000 A9 2009
2009 04 9.
200A 85 2000
2000 A9 C
                  ';'

0011

               '';
                 ';,;
                   '

0021

         ,;;
        ;',';',;;
        '''  '''

          ';'  ,'
              ';'

0031          ''
           ,   ';
          ;;  ;,,;
          '    ''

            ,,,
           ';,,'

0041

               '';
        ,;   ,'
        ''  '

        ,';
        ;;
        ',';
```

```
2000                  LIFE   ORG    $2000
2000                  CPL    *      $0020    CHARACTERS PER LINE
2000                  LIS    *      $0021    LINES IN SCREEN
2000                  CENT   *      $0022    CENTER OF POND
2000                  ADR    *      $0024    POINT TO PREV. LINE
2000                  POINT  *      $0026    POINT TO CURRENT LINE
2000                  BEFORE *      $0028    POINT TO LINE BEFORE POND
2000                  POND   *      $002A    POINT TO START OF POND
2000                  LAST   *      $002C      POINT TO LAST LINE IN POND
2000                  BUFF   *      $002E    POINT TO BUFFER
2000                  GC     *      $0030    GENERATION COUNT
2000                  GBP    *      $0032    GENERATIONS BETWEEN PRINTS
2000                  CNTG   *      $0033    COUNT OF GENERATIONS
2000                  ACT    *      $0034    ACTIVITY
2000                  OFFS   *      $0035    OFFSETS
2000                  NN     *      $003D    NO. OF NEIGHBORS
2000                  SAVY   *      $003E
2000 A9 20                   LDAIM $20       SET 32
2002 85 20                   STA   CPL       CHARS./LINE
2004 A2 10                   LDXIM $10       SET 16
2006 86 21                   STX   LIS       LINES IN SCREEN
2008 A9 00                   LDAIM $00      SET ZERO
200A 85 32                   STA   GBP       GENERATIONS BETWEEN PRINTS
200C A9 97                   LDAIM END
200E 85 28                   STA   BEFORE
2010 A9 22                   LDAIM END   /256
2012 85 29                   STA   BEFORE +01
2014 18                      CLC
2015 A5 28                   LDA   BEFORE
2017 65 20                   ADC   CPL
2019 85 2A                   STA   POND
201B 85 24                   STA   ADR
201D A5 29                   LDA   BEFORE +01
201F 69 00                   ADCIM $00
2021 85 2B                   STA   POND   +01
2023 85 25                   STA   ADR    +01

                       SET ADDRESS POINTERS

2025 20 B0 20                JSR   MULTA
2028 A5 20                   LDA   CPL
202A 4A                      LSRA
202B 18                      CLC
202C 65 24                   ADC   ADR
202E 85 22                   STA   CENT
2030 A5 25                   LDA   ADR    +01
2032 69 00                   ADCIM $00
2034 85 23                   STA   CENT   +01
2036 A6 21                   LDX   LIS
2038 20 B0 20                JSR   MULTA
203B 18                      CLC
203C A5 24                   LDA   ADR
203E 85 2C                   STA   LAST
2040 65 20                   ADC   CPL
2042 85 2E                   STA   BUFF
2044 A5 25                   LDA   ADR    +01
LIFE
2046 85 2D                   STA   LAST   +01
2048 69 00                   ADCIM $00
204A 85 2F                   STA   BUFF   +01

                         SET OFFSETS

204C A9 00                   LDAIM $00
204E 85 35                   STA   OFFS
2050 18                      CLC
2051 65 20                   ADC   CPL
2053 85 36                   STA   OFFS   +01
2055 65 20                   ADC   CPL
2057 85 37                   STA   OFFS   +02
2059 A9 01                   LDAIM $01
205B 85 38                   STA   OFFS   +03
205D 65 20                   ADC   CPL
205F 65 20                   ADC   CPL
2061 85 39                   STA   OFFS   +04
2063 A9 02                   LDAIM $02
```

0051

0061

0071



```
2065 85 3A          STA   OFFS   +05
2067 65 20          ADC   CPL
2069 85 3B          STA   OFFS   +06
206B 65 20          ADC   CPL
206D 85 3C          STA   OFFS   +07

              MAIN STRUCTURE

206F A0 00          LDYIM $00
2071 84 33          STY   CNTG
2073 84 30          STY   GC
2075 84 31          STY   GC     +01
2077 20 F2 20       JSR   CLEAR   POND
207A 20 34 21       JSR   PLANT   SEED
207D 20 D7 20 STAR  JSR   INCG     INC. GEN. COUNT
2080 20 9E 21       JSR   SHOALL  OF POND
2083 A0 00          LDYIM $00
2085 84 34          STY   ACT
2087 20 01 22       JSR   POST    BIRTHS AND DEATHS
208A 20 5A 22       JSR   UPDATE   POND
208D A5 34          LDA   ACT     IF ACTIVITY IS
208F D0 EC          BNE   STAR    ZERO
2091 00             BRK
2092 00             BRK              HALT
2093 20 60 20       JSR   STAR   +03  SHOW POND

              LINKAGE TO KIM ROUTINES

2096 4C 3B 1E PRTBYT JMP  $1E3B
2099 84 3E    GETCH  STY  SAVY
209B 20 5A 1E        JSR  $1E5A
209E A4 3E           LDY  SAVY
20A0 60              RTS
20A1 A9 0D    CRLF   LDAIM $0D
20A3 20 A8 20        JSR  OUTCH
20A6 A9 0A           LDAIM $0A
20A8 84 3E    OUTCH  STY  SAVY
LIFE
20AA 20 A0 1E        JSR  $1EA0
20AD A4 3E           LDY  SAVY
20AF 60              RTS

              ADD CPL TO ADR  (X) TIMES

20B0 CA       MULTA  DEX
20B1 30 FC           BMI  MULTA  -01
20B3 18              CLC
20B4 A5 24           LDA  ADR
20B6 65 20           ADC  CPL
20B8 85 24           STA  ADR
20BA A9 00           LDAIM $00
20BC 65 25           ADC  ADR    +01
20BE 85 25           STA  ADR    +01
20C0 4C B0 20        JMP  MULTA

              SUBTRACT  CPL (X) TIMES FROM ADR

20C3 60              RTS
20C4 CA       SUBA   DEX
20C5 30 FC           BMI  SUBA   -01
20C7 38              SEC
20C8 A5 24           LDA  ADR
20CA E5 20           SBC  CPL
20CC 85 24           STA  ADR
20CE A5 25           LDA  ADR    +01
20D0 E9 00           SBCIM $00
20D2 85 25           STA  ADR    +01
20D4 4C C4 20        JMP  SUBA

              INCREMENT AND DISPLAY GEN. COUNT

20D7 20 A1 20 INCG   JSR  CRLF
20DA 18              CLC
20DB F8              SED
20DC A9 01           LDAIM $01
20DE 65 30           ADC  GC
```

C08]

0091

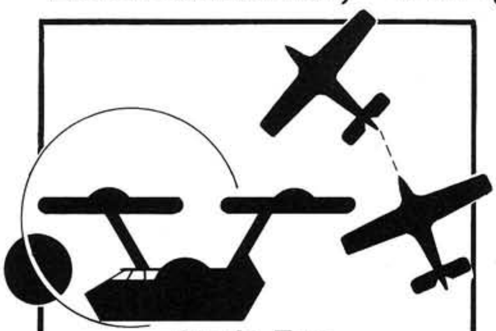
```
1510: 20E0 85 30            STA   GC
1520: 20E2 A5 31            LDA   GC      +01
1530: 20E4 69 00            ADCIM $00
1540: 20E6 85 31            STA   GC      +01
1550: 20E8 D8               CLD
1560: 20E9 20 96 20         JSR   PRTBYT
1570: 20EC A5 30            LDA   GC
1580: 20EE 20 96 20         JSR   PRTBYT
1590: 20F1 60               RTS
1600:
1610:                 CLEAR POND
1620:
1630: 20F2 20 8B 22 CLEAR   JSR   MOVE
1640: 20F5 A5 21            LDA   LIS
1650: 20F7 0A               ASLA
1660: 20F8 85 3D            STA   NN
1670: 20FA A4 20            LDY   CPL
1690: 20FC 88       CLR     DEY
LIFE
1700: 20FD 30 06            BMI   CLR     +09
1701: 20FF A9 00            LDAIM $00
1710: 2101 91 24            STAIY ADR
1720: 2103 F0 F7            BEQ   CLR
1730: 2105 A2 01            LDXIM $01
1740: 2107 20 B0 20         JSR   MULTA
1750: 210A C6 3D            DEC   NN
1760: 210C 10 EC            BPL   CLR     -02
1770: 210E 60               RTS
1780:
1790: 210F 20 A1 20 ENTRVH  JSR   CRLF
1800: 2112 A2 0B            LDXIM $0B
1810: 2114 BD 1E 21         LDAAX ENT
1820: 2117 20 A8 20         JSR   OUTCH
1830: 211A CA               DEX
1840: 211B 10 F7            BPL   ENTRVH +05
1850: 211D 60               RTS
1860: 211E 20       ENT     =     '
1870: 211F 3F               =     '?
1880: 2120 20               =     '
1890: 2121 48               =     'H
1900: 2122 2C               =     ',
1910: 2123 56               =     'V
1920: 2124 20               =     '
1930: 2125 52               =     'R
1940: 2126 45               =     'E
1950: 2127 54               =     'T
1960: 2128 4E               =     'N
1970: 2129 45               =     'E
1980: 212A 20 99 20 GET     JSR   GETCH
1990: 212D C9 38            CMPIM '8
2000: 212F 30 02            BMI   DONE
2010: 2131 A9 00            LDAIM $00
2020: 2133 60       DONE    RTS
2030:
2040:                 PLANT SEED
2050:
2060: 2134 20 0F 21 PLANT   JSR   ENTRVH
2070: 2137 20 2A 21         JSR   GET
2080: 213A F0 F8            BEQ   PLANT
2090: 213C C9 30            CMPIM '0
2100: 213E 30 F3            BMI   DONE
2110: 2140 29 07            ANDIM $07
2120: 2142 AA               TAX
2130: 2143 A5 22            LDA   CENT
2140: 2145 85 24            STA   ADR
2150: 2147 A5 23            LDA   CENT    +01
2160: 2149 85 25            STA   ADR     +01
2170: 214B 20 2A 21         JSR   GET
2180: 214E F0 E4            BEQ   PLANT
2190: 2150 C9 2D            CMPIM '-
2200: 2152 F0 05            BEQ   MINUS
2210: 2154 20 B0 20         JSR   MULTA
2220: 2157 30 03            BMI   HOR
2230: 2159 20 C4 20 MINUS   JSR   SUBA
2240: 215C A2 02    HOR     LDXIM $02
LIFE
2250: 215E 20 14 21         JSR   ENTRVH +05
2260: 2161 20 2A 21         JSR   GET
2270: 2164 C9 30            CMPIM '0
2280: 2166 30 CC            BMI   PLANT
2290: 2168 29 07            ANDIM $07
2300: 216A 85 3D            STA   NN
2310: 216C 20 2A 21         JSR   GET
2320: 216F F0 C3            BEQ   PLANT
2330: 2171 C9 2D            CMPIM '-
2340: 2173 F0 16            BEQ   MIN
2350: 2175 18               CLC
2360: 2176 A5 24            LDA   ADR
2370: 2178 65 3D            ADC   NN
2380: 217A 85 24            STA   ADR
2390: 217C A5 25            LDA   ADR     +01
2400: 217E 69 00            ADCIM $00
2410: 2180 85 25            STA   ADR     +01
2420: 2182 A9 01            LDAIM $01
2421: 2184 A0 00            LDYIM $00
2430: 2186 91 24            STAIY ADR
2440: 2188 4C 34 21         JMP   PLANT
2450: 218B 38       MIN     SEC
2460: 218C A5 24            LDA   ADR
2470: 218E E5 3D            SBC   NN
2480: 2190 85 24            STA   ADR
2490: 2192 A5 25            LDA   ADR     +01
2500: 2194 E9 00            SBCIM $00
2510: 2196 4C 82 21         JMP   MIN     -09
2520:
2530:                 SHOW ALL OF POND
2540:
2550: 2199 A5 32            LDA   GBP
2560: 219B 85 33            STA   CNTG
2570: 219D 60               RTS
2580: 219E C6 33    SHOALL  DEC   CNTG
2590: 21A0 10 FB            BPL   SHOALL -01
2600: 21A2 20 8B 22         JSR   MOVE
2610: 21A5 A5 21            LDA   LIS
2620: 21A7 85 3D            STA   NN
2630: 21A9 A2 01    SHO     LDXIM $01
2640: 21AB 20 B0 20         JSR   MULTA
2650: 21AE C6 3D            DEC   NN
2660: 21B0 F0 E7            BEQ   SHOALL -05
2680: 21B2 A4 20            LDY   CPL
2690: 21B4 B1 24            LDAIY ADR
2700: 21B6 D0 04            BNE   SHOA
2710: 21B8 A9 20            LDAIM $20
2720: 21BA 10 02            BPL   SHOA    +02
2730: 21BC A9 27    SHOA    LDAIM $27
2740: 21BE 91 2E            STAIY BUFF
2750: 21C0 88               DEY
2760: 21C1 D0 F1            BNE   SHOA    -08
2770: 21C3 A2 01            LDXIM $01
2780: 21C5 20 B0 20         JSR   MULTA
2790: 21C8 A4 20            LDY   CPL
2800: 21CA B1 24    SHOW    LDAIY ADR
LIFE
2810: 21CC F0 0E            BEQ   SHOWB   +02
2820: 21CE B1 2E            LDAIY BUFF
2830: 21D0 C9 20            CMPIM $20
2840: 21D2 F0 04            BEQ   SHOWB   -02
2850: 21D4 A9 3B            LDAIM ';
2860: 21D6 10 02            BPL   SHOWB
2870: 21D8 A9 2C            LDAIM ',
2880: 21DA 91 2E    SHOWB   STAIY BUFF
2890: 21DC 88               DEY
2900: 21DD D0 EB            BNE   SHOW
2910: 21DF A4 20            LDY   CPL
2920: 21E1 B1 2E            LDAIY BUFF
2930: 21E3 C9 20            CMPIM $20
2940: 21E5 D0 03            BNE   SHOWA
2950: 21E7 88               DEY
2960: 21E8 D0 F7            BNE   SHOWB   +07
2970: 21EA C8       SHOWA   INY
2980: 21EB A9 0D            LDAIM $0D
2990: 21ED 91 2E            STAIY BUFF
```

```
3000: 21EF A0 00              LDYIM $00
3010: 21F1 20 A1 20           JSR   CRLF
3020: 21F4 C8         SHOP    INY
3030: 21F5 B1 2E              LDAIY BUFF
3040: 21F7 C9 0D              CMPIM $0D
3050: 21F9 F0 AE              BEQ   SHO
3060: 21FB 20 AB 20           JSR   OUTCH
3070: 21FE 4C F4 21           JMP   SHOP
3080:
3090:                 POST BIRTHS AND DEATHS
3100:
3110: 2201 20 8B 22   POST    JSR   MOVE
3120: 2204 20 79 22           JSR   INCP
3130: 2207 A9 00              LDAIM $00
3140: 2209 85 3D              STA   NN
3150: 220B A2 07              LDXIM $07
3160: 220D B5 35      NBR     LDAZX OFFS
3170: 220F A8                 TAY
3180: 2210 B1 24              LDAIY ADR
3190: 2212 F0 04              BEQ   NB
3200: 2214 30 02              BMI   NB
3210: 2216 E6 3D              INC   NN
3220: 2218 CA         NB      DEX
3230: 2219 10 F2              BPL   NBR
3240: 221B A0 00              LDYIM $00
3250: 221D A5 3D              LDA   NN
3260: 221F C9 02              CMPIM $02
3270: 2221 30 20              BMI   DEATH
3280: 2223 C9 03              CMPIM $03
3290: 2225 F0 24              BEQ   BIRTH
3300: 2227 10 1A              BPL   DEATH
3310: 2229 20 73 22   POSTA   JSR   INCA
3320: 222C D0 D9              BNE   POST   +06
3330: 222E 20 8B 22   EDGE    JSR   MOVE
3340: 2231 A5 21              LDA   LIS
3350: 2233 0A                 ASLA
3360: 2234 85 3D              STA   NN
LIFE

3370: 2236 98                 TYA
3380: 2237 91 24              STAIY ADR
3390: 2239 A2 01              LDXIM $01
3400: 223B 20 B0 20           JSR   MULTA
3410: 223E C6 3D              DEC   NN
3420: 2240 10 F4              BPL   EDGE   +08
3430: 2242 60                 RTS
3440: 2243 B1 26      DEATH   LDAIY POINT
3450: 2245 F0 E2              BEQ   POSTA
3460: 2247 A9 02              LDAIM $02
3470: 2249 10 06              BPL   BIRTH  +06
3480: 224B B1 26      BIRTH   LDAIY POINT
3490: 224D D0 DA              BNE   POSTA
3500: 224F A9 FF              LDAIM $FF
3510: 2251 91 26              STAIY POINT
3520: 2253 E6 34              INC   ACT
3530: 2255 F0 FC              BEQ   BIRTH  +08
3540: 2257 4C 29 22           JMP   POSTA
3550:
3560:                 UPDATE POND
3570:
3580: 225A 20 8B 22   UPDATE  JSR   MOVE
3590: 225D B1 26              LDAIY POINT
3600: 225F 30 08              BMI   ADULT
3610: 2261 C9 02              CMPIM $02
3620: 2263 30 08              BMI   ADULT  +04
3630: 2265 A9 00      BURY    LDAIM $00
3640: 2267 F0 02              BEQ   ADULT  +02
3650: 2269 A9 01      ADULT   LDAIM $01
3660: 226B 91 26              STAIY POINT
3670: 226D 20 79 22           JSR   INCP
3680: 2270 D0 EB              BNE   UPDATE +03
3690: 2272 60                 RTS
3700: 2273 E6 24      INCA    INC   ADR
3710: 2275 D0 02              BNE   INCP
3720: 2277 E6 25              INC   ADR    +01
3730: 2279 E6 26      INCP    INC   POINT
3740: 227B D0 02              BNE   INCP   +06
3750: 227D E6 27              INC   POINT  +01
3760: 227F A5 27              LDA   POINT  +01
3770: 2281 C5 2D              CMP   LAST   +01
3780: 2283 30 05              BMI   MOVE   -01
3790: 2285 38                 SEC
3800: 2286 A5 26              LDA   POINT
3810: 2288 E5 2C              SBC   LAST
3820: 228A 60                 RTS
3830: 228B A2 03      MOVE    LDXIM $03
3840: 228D B5 28              LDAZX BEFORE
3850: 228F 95 24              STAZX ADR
3860: 2291 CA                 DEX
3870: 2292 10 F9              BPL   MOVE   +02
3880: 2294 A0 00              LDYIM $00
3890: 2296 60                 RTS
3900: 2297 00         END     =     $00
3910: 2298 00                 =     $00

SYMBOL TABLE 3300 3450
ACT     0034   ADR     0024 ADULT  2269   BEFORE 0028
BIRTH   224B   BUFF    002E BURY   2265   CENT   0022
CLEAR   20F2   CLR     20FC CNTG   0033   CPL    0020
CRLF    20A1   DEATH   2243 DONE   2133   EDGE   222E
END     2297   ENTRVH  210F ENT    211E   GBP    0032
GC      0030   GETCH   2099 GET    212A   HOR    215C
INCA    2273   INCG    20D7 INCP   2279   LAST   002C
LIFE    2000   LIS     0021 MINUS  2159   MIN    21BB
MOVE    228B   MULTA   20B0 NB     2218   NBR    220D
NN      003D   OFFS    0035 OUTCH  20AB   PLANT  2134
POINT   0026   POND    002A POST   2201   POSTA  2229
PRTBYT  2096   SAVY    003E SHOA   21BC   SHOALL 219E
SHOP    21F4   SHOW    21CA SHOWA  21EA   SHOWB  21DA
SHO     21A9   STAR    207D SUBA   20C4   UPDATE 225A
```

μ

Natural Organic Apple Software

Educational, intriguing and challenging. . . naturally!



Apple Fun

We've taken five of our most popular programs and combined them into one tremendous package full of fun and excitement. This disk-based package now offers you these great games:

Mimic—How good is your memory? Here's a chance to find out! Your Apple will display a sequence of figures on a 3×3 grid. You must respond with the exact same sequence, within the time limit.

There are five different, increasingly difficult versions of the game, including one that will keep going indefinitely. Mimic is exciting, fast paced and challenging—fun for all!

Air Flight Simulation—Your mission is to take off and land your aircraft without crashing. You're flying blind: on instruments only.

You start with a full tank of fuel, which gives you a maximum range of approximately 50 miles. The computer will constantly display updates of your air speed, compass heading and altitude. Your most important instrument is the Angle of Ascent/Bank Indicator. It will tell if the plane is climbing or descending and whether banking into a right of left turn.

After you've acquired a few hours flying time, you can try flying a course against a map or doing aerobatic maneuvers. Get a little more flight time under your belt and the sky's the limit!

Colormaster—Test your powers of deduction as you try to guess the secret color code in this Mastermind-type game. There are two levels of difficulty, and three options of play to vary your games. Not only can you guess the computer's color code, but it will guess yours! It will also serve as referee in a game between two human opponents. Can you make and break the color code. . .?

Star Ship Attack—Your mission is to protect our orbiting food station satellites from destruction by an enemy star ship. You must capture, destroy or drive off the attacking ship. If you fail, our planet is doomed.

Trilogy—This exciting contest of logic has its origins in the simple game of tic-tac-toe. The object of the game is to place three of your colors in a row into the delta-like, multi-level display. The rows may be horizontal, vertical, diagonal and wrapped around, through the "third dimension". Your Apple (or human opponent) will be trying to do the same, and there are many paths to victory. You can even have your Apple play against itself!

Minimum system requirements are an Apple II or Apple II Plus computer with 32K of memory and one minidisk drive. Mimic requires Applesoft in ROM, all others run in RAM or ROM Applesoft.

Order No. 0161AD $19.95

Paddle Fun

This new Apple disk package requires a steady eye and a quick hand at the game paddles! We've included four different games to challenge and amuse you. They include:

Invaders—You must destroy an invading fleet of 55 flying saucers while dodging the carpet of bombs they drop. Keep a wary eye for the mother ship directing the incursion. Your bomb shelters will help you—for a while. Our version of a well known arcade game! Requires Applesoft in ROM.

Howitzer—This is a one or two person game in which you must fire upon another howitzer position. This program is written in HIGH-RESOLUTION graphics using different terrain and wind conditions each round to make this a demanding game. The difficulty level can be altered to suit the ability of the players. Requires Applesoft in ROM.

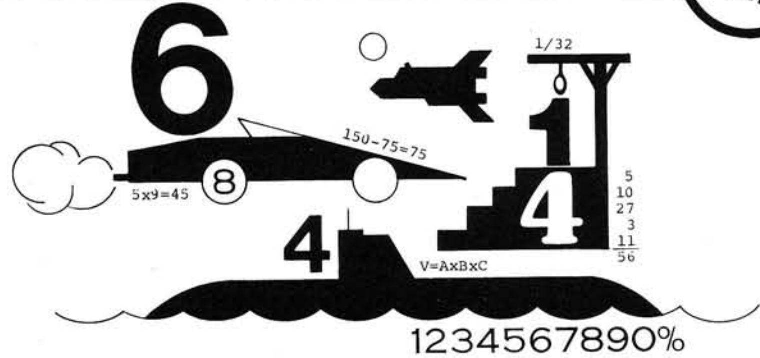
Space Wars—This program has three parts: (1) Two flying saucers meet in laser combat—for two players, (2) two saucers compete to see which can shoot out the most stars—for two players, and (3) one saucer shoots the stars in order to get a higher rank—for one player only. Requires Applesoft.

Golf—Whether you win or lose, you're bound to have fun on our 18 hole Apple golf course. Choose your club and your direction and hope to avoid the sandtraps. Losing too many strokes in the water hazards? You can always increase your handicap. Get off the tee and onto the green with Apple Golf. One of its nicest features is you'll never need to cancel a golf date due to rain. Requires Applesoft.

The minimum system requirement for this package is an Apple II or Apple II Plus computer with 32K of memory and one minidisk drive.

Order No. 0163AD $19.95

ON DISK



Math Fun

Change an Apple computer into a mathematics tutor and change boredom into enthusiasm with the Math Fun package. Using the technique of immediate positive reinforcement, students can improve their math skills while playing a game with:

Hanging—A little man is walking up the steps to the hangman's noose. But YOU can save him by answering the problems posed by the computer. The program uses decimal math problems. Each correct answer will move the man down the steps and cheat the hangman.

Spellbinder—You are a magician competing against a computerized wizard. In order to cast death clouds, fireballs and other magic spells on him, you must correctly answer questions about using fractions.

Whole Space—Pilot your space craft to attack the enemy planet. Each time you give a correct answer to the whole number problems posed by the computer, you move your ship. But for every wrong answer, the enemy gets a chance to fire at you.

Car Jump—Make your stunt car jump the ramps. Each correct answer will increase the number of buses your car must jump over. These problems involve calculating the areas of different geometric figures.

Robot Duel—Fire your laser cannon at the computer's robot. If you give the correct answer to problems on calculating volumes, your robot can shoot at his opponent. If you give the wrong answer, your shield power will be depleted and the computer's robot can shoot at yours.

Sub Attack—Practice using percentages as you maneuver your sub into the harbor. A correct answer lets you move your sub and fire at the enemy fleet.

All of these programs run in Applesoft BASIC, except Whole Space, which requires Integer BASIC.

Order No. 0160AD $19.95

TO ORDER: Look for these programs at the dealer nearest you. If your store doesn't stock Instant Software send your order with payment to: Instant Software, Order Dept., Peterborough, N.H. 03458 (add $1.00 for handling) or call toll-free 1-800-258-5473 (VISA, MC and AMEX accepted).

Instant Software™

Prices subject to change without notice.

PETERBOROUGH, N.H. 03458
603-924-7296

GALAXY SPACE WAR I

Galaxy Space War 1 (WAR1) is a game of strategy in which the player has complete control of his space fleet's tactical maneuvers. Each fleet battles its way toward the opponents galaxy in an attempt to destroy it and win the war. WAR1 simulates the actual environment encountered in a space war between two galaxies. Optimum use is made of Apple's high resolution graphics (HIRES) and colors in displaying the twinkling stars universe, the colored ships of each fleet, long range sensors colored illuminations, and the alternating blinking colors used in battles between ships. Complementing HIRES are the sounds of war produced by Apple's speaker.

WAR1 is played between Apple and a player or between two players. You may play with total knowledge of each others fleet or only ships sensor knowledge of the opponents fleet. Each player builds his starting fleet and adds to it during the game. This building process consists of creating the size and shape of each ship, positioning it, and then allocating the total amount of energy fcr each ship.

During a player's turn he may dynamcially allocate his ships total energy between his screen/detection and attack/move partitions. The percentage of the total energy allocated to each partition determines its characteristics. The screen/detection partition determines how much energy is in a ship's screens and the detection sector range of its short range sensors. The attack/move determines the amount of energy the ship can attack with, its attack sector range, and the number of sectors it can move in normal or hyperspace.

When an enemy ship is detected by short range sensors, it is displayed on the universe and a text enemy report appears. The report identifies the ship, its position, amount of energy in its screens, probable attack and total energy, a calculated detection/attack/move range, and size of the ship. Also shown is the number of days since you last knew these parameters about the ship. When a ship's long range sensor probes indicate the existence of an enemy presence at a sector in space, this sector is illuminated on the universe.

An enemy ship is attacked and destroyed with attack energy. If your attack energy breaks through his screens, then his attack energy is reduced by two units of energy for every unit you attack with. A text battle report is output after each attack. The program maintains your ship's data and the latest known data about each enemy ship. You may show either data in text reports or display the last known enemy positions on the universe. You can also get battle predictions between opposing ships. The text output calculates the amount of energy required to destroy each ship for different energy allocations.

APPLE® II, 48K, APPLESOFT
ROM CARD, DISK II DOS 3.2
WAR1 DISK & MANUAL ...$39.95
(CA residents add 6% sales tax)
Write or call for more information

GALAXY
DEPT. MI3
P.O. BOX 22072
SAN DIEGO, CA 92122
(714) 452-1072

Decision Systems

Decision Systems
P.O. Box 13006
Denton, TX 76203

INDEXED FILES — **$50**

ISAM-DS is an integrated set of routines for the creation and manipulation of indexed files. **ISAM-DS** provides capabilities comparable to those on large mainframes. You can rapidly retrieve records by key value or partial key value (retrieves any record in a 200 record file, 60 char/record, in less than 3 seconds compared to a maximum of 38 seconds for a **DOS** sequential file). Files never have to be reorganized. Duplicate key values may be used. Records may also be retrieved in sequence. **ISAM-DS** routines are easily integrated into Applesoft programs — they use less than **3K RAM** plus an index table.

Requires: Disk, Applesoft

STRUCTURED BASIC — **$35**

PBASIC-DS is a sophisticated preprocessor for structured **BASIC**. Now you can gain the power of **PASCAL**-like logic structures at a fraction of the cost. Use all regular **BASIC** statements plus 14 commands and 11 new statements/structures (**WHILE, UNTIL, CASE**, etc.). **PBASIC-DS** can be used to develop **INTEGER** or **APPLESOFT** programs. It is a great way to learn and use structured logic concepts.

Requires: Disk, Applesoft (48K ROM)

DATA ENTRY — **$25**

FORM-DS is a system of programs and routines that assist in the entry, editing and display of data. Describe screen formats by simply typing them on the screen. Automatic range tests for input data. Display edited numeric values with commas inserted, etc. Dump the screen contents to a printer. Routines are easily incorporated into Applesoft programs. Documentation included.

Requires: Disk, Applesoft (32K ROM)

(Texas residents add 5% tax)
(Add $4.00 for Foreign Mail)

*Apple II is a registered trademark of the Apple Computer Co.

Apple Owners:

DEPRECIATION
PROGRAM

- ★ 5 DEPRECIATION RATES
- ★ UP TO 99 YR TERM
- ★ RECORDS UP TO 600 ITEMS ON DISK
- ★ UP TO $1 MILLION FOR EACH ITEM
- ★ REPORTS EACH MONTH, QUARTER, OR ANNUALLY
- ★ BONUS DEPR., INVESTMENT CREDIT
- ★ PRO-RATES DEPRECIATION
- ★ UPDATE RECORDS EACH YEAR
- ★ EQUIPMENT INVENTORY
- ★ FISCAL YEAR BASED
- ★ CONVERT METHODS ANY TIME
- ★ AN ACCOUNTANTS DREAM

INTRODUCTORY OFFER

APPLESOFT ....32K MIN.. ~~$225.00~~ $150.00
HANDBOOK............$5.00

VISA & M/C USERS — CALL
509-943-9004

MONEYDISK
516 WELLSIAN WAY
RICHLAND, WA 99352

APPLE IS A REGISTERED TRADEMARK
OF APPLE COMPUTER INC.
WA Residents, add 5% sales tax
DEALER INQUIRIES INVITED

ASCII EXPRESS II
By Bill Blue

THE MOST COMPLETE COMMUNICATIONS PACKAGE
FOR THE APPLE II AVAILABLE

This professional terminal system includes:

- Full upper/lower case support
- Upload/download files to ANY type of online computer
- Built-in line editor for files
- Keyboard macros for efficient operation

Recommended system requirements include a 48K APPLE II with disk, FP ROM or language card, and Micromodem or Comm. card.

$59.95 plus 6% for Calif. residents

Available from your local computer store or:

southwestern data systems
P.O. BOX 582-M • SANTEE, CA 92071
(714) 562-3670

Step and Trace for the APPLE II Plus

If you miss the Step and Trace of the original APPLE II on your new APPLE II Plus, here is all you need to restore it.

Craig Peterson
1743 Centinela Avenue #102
Santa Monica, CA 90404

Apple Computer's new APPLE II Plus is a pretty good machine. It has improved editing features over those of the standard APPLE II and a better cursor control and stop list feature. And it's really nice to fire up the machine and be right in BASIC or DOS, or better yet, to be in the middle of a turn-key type program.

Furthermore, Applesoft BASIC is a standard feature, and I'm partial to it over Integer BASIC. But all of these improvements didn't come for free. There's only so much room in the ROM monitor, and certain of its features had to be sacrificed to make room for the new additions. As a result, the machine language step-and-trace capabilities of the older APPLE II ended up on the cutting room floor.

A lot of people will probably never miss step and trace. Unless you are into assembly language programming, you probably don't need them. But if you do any assembly language programming, step and trace can be invaluable. They allow you to step through each machine language instruction, displaying all of the 6502 registers as you go along, so you can find any errors that might exist in the program, or even just see how the program works. Step does this one instruction at a time, and trace does it continuously, without stopping (unless a break instruction is encountered).

Well, fear not, APPLE II Plus owners, Step-n-Trace is here. The Step-n-Trace (S&T) program essentially just adds the step-and-trace functions to the existing monitor of your APPLE II Plus. The operation and use of the monitor is identical to that of the original APPLE monitor. Type a hex address followed by one or more 'S's, to take steps through a program from that address. To trace, type a hex address followed by a 'T', to begin tracing from that address.

An improved feature of S&T over the original APPLE trace is that all you have to do is press any key (for example, the space bar) to stop the trace. To continue tracing, type a 'T', and trace will continue from where it stopped. Or you can type an 'S' to take only one step. The prompt character used for S&T is an inverse '*' so you can distinguish it from the normal APPLE monitor. S&T also includes all of the normal monitor commands in addition to step and trace. In fact, it actually uses many parts of the existing monitor to do its work.

To use Step-n-Trace, first load it and then type 'CALL 768', or 'BRUN' it from your disk, if you have one. You will then have all of the monitor commands at your disposal, including step and trace. To get out of the program, just press 'RESET' on your APPLE II Plus, or use 'CTRL', 'C', or 'CTRL' 'B' and you will end up in BASIC.

Since the program resides in hex address $300 to $3E9, it loads over some of the DOS address pointers from $3DO to $3E9. Generally, this doesn't cause any problems for me. However, this can be avoided by moving it to some other area of memory; but the jump addresses in lines 590, 650, 730, 1100, 1580, and 1590 will have to be revised accordingly. The assembler listing for S&T makes use of most of the same labels as the APPLE monitor to make it easier to relate what's happening with the old monitor.

At this point, I should mention that the step-and-trace functions from the same problems as the original APPLE monitor, in that, under certain conditions, the stack register will be displayed with an incorrect value. When this happens, for example, after JSR or RTS, the display will be corrected after the next instruction. Also, if the program manipulates the stack with the use of TXS instructions, the actual operation will probably be incorrect. Lastly, with DOS in effect, when a program is traced through the changing of an I/O hook (usually $36 or $37) the program trace will lock up because the output will have a partially incorrect jump indirect address, and your trace will fall off the edge of the earth. The frailties mentioned above are not nearly as restrictive as they may seem. All in all, S&T is a useful utility.

For those of you who have read thus far, but don't really plan on doing any assembly language programming, here is how Applesoft works. First load Step-n-Trace and then enter the following BASIC program:

```
10 CALL 768: PRINT "HELLO"
20 END
```

Next type 'RUN', and you will be rewarded with the sound of the bell and an inverse '*' prompt character, telling you that you're in S&T. Next type 'FF58S'. From now on, each 'S' you type will step you through the operations of Applesoft. The first 'S' should display 'D823- 4C D2 D7 JMP $D7D2' on the screen, followed by the contents of the registers. This is the running return to Applesoft. As you 'S'tep or 'T'race through the instructions, you will see the colon ($3A), the print command token ($BA), the quotation ($22), the characters of the word 'HELLO' ($48,45,4C,4C,4F) and more pass through the A (accumulator) register, as Applesoft analyzes your program line.

With some study you'll begin to understand what Applesoft is doing. With some effort, you can actually find where the subroutines are located for the 'SIN', 'SQR', or any other function you're interested in. All of this is accomplished with the help of S&T.

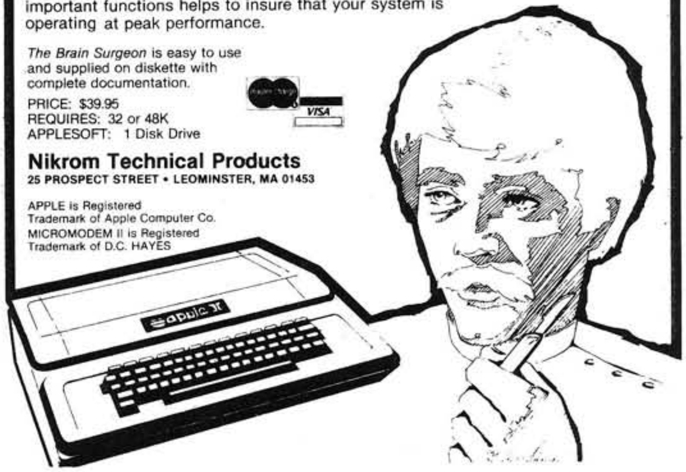
So, if you're doing any assembly language work on an APPLE II Plus, S&T can be of great help. If you're just interested in seeing how things actually run inside your APPLE, Step-n-Trace can open a lot of interesting doors. Anyway, have fun, and if you find out anything interesting, write about it.

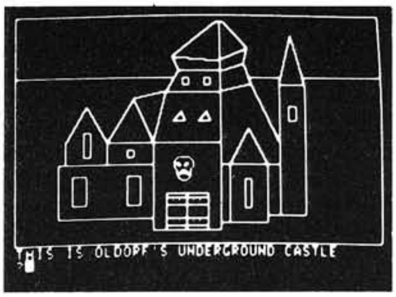
```
              :********************************
              :*          STEP-N-TRACE        *
              :*               BY             *
              :*         CRAIG PETERSON       *
              :*          FEBRUARY 1980       *
              :*                              *
              :*A PROGRAM TO FURNISH THE APPLE*
              :* II PLUS WITH THE STEP & TRACE*
              :* CAPABLILITIES OF THE STANDARD*
              :* APPLE II                     *
              :********************************
              :
              RTNL .DL 002C     RETURN ADRES LO
              RTNH .DL 002D     RETURN ADRES HI
              LGTH .DL 002F     LNGTH/DSPLACMNT
              PRMP .DL 0033     PROMPT CHARACTR
              YSAV .DL 0034     PLACE TO SAVE Y
              PCL  .DL 003A     PROGRAM CNTR LO
              PCH  .DL 003B     PROGRAM CNTR HI
              XQT  .DL 003C     USR INSTRUCTION
              STAT .DL 0048     PROC STATUS REG
              KBRD .DL C000     KEYBOARD REGSTR
              INSD .DL F882     DSPLAY PGM CNTR
              DISA .DL F8D0     DISASEMBL INSTR
              ADJ2 .DL F954     ADJUST PC - 2
              ADJ3 .DL F956     ADJUST PC - 3
              REGD .DL FAD7     DSPLAY USR REGS
              RGDS .DL FADA     DISP REGS-NO CR
              GETL .DL FD67     GET INPUT LINE
              BL1  .DL FE00     BLANK ROUTINE
              A1PC .DL FE75     COPY A1 TO PC
              BELL .DL FF3A     RING THE BELL
              RSTR .DL FF3F     RESTOR USR REGS
              SAVE .DL FF4A     SAVE USER REGS
              GETN .DL FFA7     GET ITEM,NONHEX
              TSUB .DL FFBE     PUSH & GOTO SUB
              TSB1 .DL FFC5     HANDLE THE MODE
              ZMOD .DL FFC7     ZERO THE MODE
              CHRT .DL FFCC     CHARACTER TABLE
              :
                   .OR 0300
              :
0300 D8       STRT CLD          SET HEX MODE
0301 203AFF        JSR BELL     RING THAT CHIME
0304 A92A     CONT LDA '*'      LOAD INVERS '*'
0306 8533          STA *PRMP    & STOR IN PRMPT
0308 2067FD        JSR GETL     READ A LINE
030B 20C7FF        JSR ZMOD     SET MODE & Y =0
030E 20A7FF   NXTI JSR GETN     GET ITEM,NONHEX
0311 8434          STY *YSAV     CHAR IN A-REG
0313 C9EC     TRYS CMP 0EC      IS IT STEP?
0315 F00B          BEQ ENT2     IF=STEP,GO ENT2
0317 C9ED     TRYT CMP 0ED      IS IT TRACE?
0319 D00F          BNE TRCR     IF<>TRACE,TRYCR
031B AD00C0        LDA KBRD     WAS KEY PRESSD?
031E 3024          BMI AGIN     KEY ON,-->AGIN
0320 C634          DEC *YSAV    MAKES STEP RPT
0322 20C7FF   ENT2 JSR ZMOD     ENTRY FOR STEP
0325 204903        JSR STPZ     GO STEP OUT
0328 101A          BPL AGIN     RTN TO INP LINE
032A C9C6     TRCR CMP 0C6      IS IT A CR?
032C D009          BNE MCMD     IF<>CR,TRY MCMD
032E 20C5FF        JSR TSB1
0331 2000FE        JSR BL1      HNDL CR AS BLNK
0334 4C0403        JMP CONT     RETURN TO CONT
0337 A017     MCMD LDY 17       TRY MONITR CMDS
0339 88       CHRS DEY          SERCH MON CHARS
033A 30C4          BMI STRT     NOT FND,GO STRT
033C D9CCFF        CMP CHRT,Y   CMP WITH TABLE
033F D0F8          BNE CHRS     NOT FND,->CHRS
0341 20BEFF        JSR TSUB     FND, CALL SUB
0344 A434     AGIN LDY *YSAV    RESTORE Y
0346 4C0E03        JMP NXTI     GET NXT COMMAND
0349 2075FE   STPZ JSR A1PC     ADR TO PC
034C 20D0F8   STEP JSR DISA     TAKE ONE STEP
034F 68            PLA          ADJUST TO USER
0350 852C          STA *RTNL     STACK AND SAVE
0352 68            PLA           RTN ADR
0353 852D          STA *RTNH
0355 A208          LDX 08
0357 BDE103   XQIN LDA INM1,X   INIT XEQ AREA
035A 953C          STA *XQT ,X
035C CA            DEX
035D D0F8          BNE XQIN
035F A13A          LDA (PCL ,X) USR OPCODE BYTE
0361 F02C          BEQ XBRK     SPESHL IF BREAK
0363 A42F          LDY *LGTH    LGTH FROM DASSY
0365 C920          CMP 20
0367 F043          BEQ XJSR     HANDLE JSR,RTS,
0369 C960          CMP 60        JMP,JMP( ),
036B F02F          BEQ XRTS      & RTI SPECIAL
036D C94C          CMP 4C
036F F046          BEQ XJMP
0371 C96C          CMP 6C
0373 F043          BEQ XJAT
0375 C940          CMP 40
0377 F01F          BEQ XRTI
0379 291F          AND 1F
037B 4914          EOR 14
037D C904          CMP 04       COPY USR INSTR
037F F002          BEQ XQ2       TO XEQ AREA
0381 B13A     XQ1  LDA (PCL ),Y
0383 993C00   XQ2  STA XQT ,Y
0386 88            DEY
0387 10F8          BPL XQ1
0389 203FFF        JSR RSTR     RESTOR USR REGS
038C 4C3C00        JMP XQT      XEQ USER OP
038F 2082F8   XBRK JSR INSD     PRINT USER PC
0392 20DAFA        JSR RGDS      AND REGS
0395 4C0003        JMP STRT     THEN GO TO STRT
0398 18       XRTI CLC
0399 68            PLA          SIMULATE RTI
039A 8548          STA *STAT
039C 68       XRTS PLA          RTS SIMULATION
039D 853A          STA *PCL
039F 68            PLA
03A0 853B     PCN2 STA *PCH
03A2 A52F     PCN3 LDA *LGTH    UPDAT PC BY LEN
03A4 2056F9        JSR ADJ3
03A7 843B          STY *PCH
03A9 18            CLC
03AA 9014          BCC NEWP
03AC 18       XJSR CLC
03AD 2054F9        JSR ADJ2     UPDATE PC AND
03B0 AA            TAX           PUSH ONTO STAK
03B1 98            TYA           FOR JSR
03B2 48            PHA           SIMULATION
03B3 8A            TXA
03B4 48            PHA
03B5 A002          LDY 02
03B7 18       XJMP CLC
03B8 B13A     XJAT LDA (PCL ),Y
03BA AA            TAX          LOAD PC FOR JMP
03BB 88            DEY           & (JMP)
03BC B13A          LDA (PCL ),Y  SIMULATION
03BE 863B          STX *PCH
03C0 853A     NEWP STA *PCL
03C2 B0F3          BCS XJMP
```

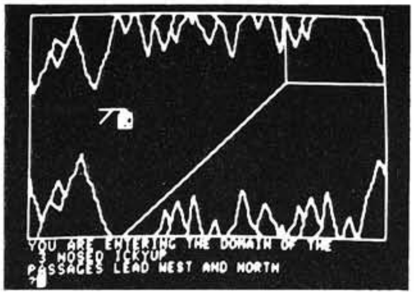
```
03C4  A52D     RTNJ LDA *RTNH
03C6  48            PHA
03C7  A52C          LDA *RTNL
03C9  48            PHA
03CA  4CD7FA        JMP REGD        DISPLAY USR REG
03CD  18       BRAN CLC             BRANCH TAKEN,
03CE  A001          LDY 01           AD LEN+2 TO PC
03D0  B13A          LDA (PCL ),Y
03D2  2056F9        JSR ADJ3
03D5  853A          STA *PCL
03D7  98            TYA
03D8  38            SEC
03D9  B0C5          BCS PCN2
03DB  204AFF   NBRN JSR SAVE        NORML RTRN AFTR
03DE  38            SEC              EXQING USER OP
03DF  B0C1          BCS PCN3        GO UPDATE PC
03E1  EA       INM1 NOP
03E2  EA       INIT NOP
03E3  EA            NOP             DUMMY FILL FOR
03E4  4CDB03        JMP NBRN         XEQ AREA
03E7  4CCD03        JMP BRAN
                    .EN
```

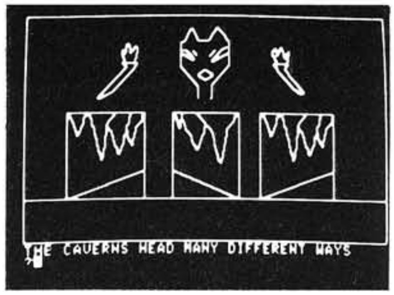
```
SYMBOL TABLE    BELL  FF3A      XQIN  0357
RTNL  002C      RSTR  FF3F      XQ1   0381
RTNH  002D      SAVE  FF4A      XQ2   0383
LGTH  002F      GETN  FFA7      XBRK  038F
PRMP  0033      TSUB  FFBE      XRTI  0398
YSAV  0034      TSB1  FFC5      XRTS  039C
PCL   003A      ZMOD  FFC7      PCN2  03A0
PCH   003B      CHRT  FFCC      PCN3  03A2
XQT   003C      STRT  0300      XJSR  03AC
STAT  0048      CONT  0304      XJMP  03B7
KBRD  C000      NXTI  030E      XJAT  03B8
INSD  F882      TRYS  0313      NEWP  03C0
DISA  F8D0      TRYT  0317      RTNJ  03C4
ADJ2  F954      ENT2  0322      BRAN  03CD
ADJ3  F956      TRCR  032A      NBRN  03DB
REGD  FAD7      MCMD  0337      INM1  03E1
RGDS  FADA      CHRS  0339      INIT  03E2
GETL  FD67      AGIN  0344
BL1   FE00      STPZ  0349
A1PC  FE75      STEP  034C
```

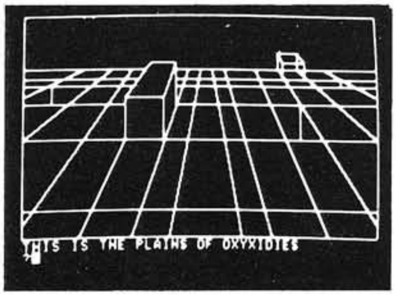
μ

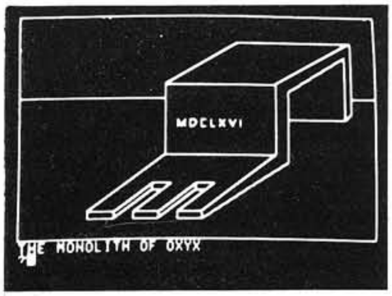


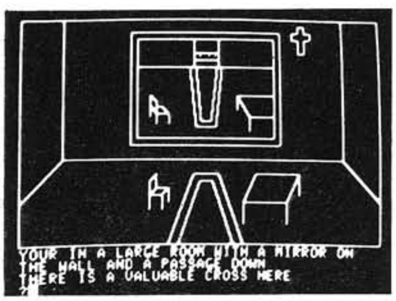


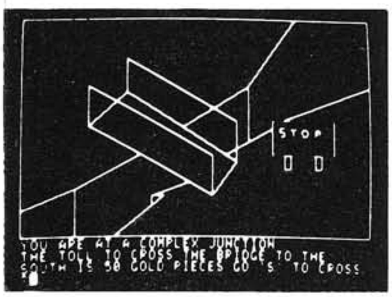












AIM 65 File Operations: Writing Text Files with BASIC

The value of BASIC is greatly enhanced with the capability of writing text files. The techniques and programs required are presented here.

Christopher J. Flynn
2601 Claxton Drive
Herndon, VA 22070

In an article published in MICRO, July 1980 (26:61), I presented a subroutine which made AIM 65 text files accessible to BASIC. The capability to read text files with BASIC has satisfied many of our requirements. I also hope that the subroutine has been of help to other AIM 65 users as well.

There are, however, many applications which require the capability to write text files. Therefore, I have developed a second machine language subroutine to meet this need. This subroutine provides a means for creating AIM 65 text files directly from BASIC. Using this subroutine, you can store any kind of data on tape-character strings or numbers. With this subroutine you can also use BASIC to write other BASIC programs! In fact, our sample program will do just that.

As was the case with the earlier text file input subroutine, the text file output subroutine is both ROMable and completely position independent. Don't be surprised if you see sections of code from the input subroutine duplicated in the output subroutine. I have tried to emphasize ease of use above other factors.

The text file input and output subroutines handle all the file operations that will normally be required. However, I must point out one restriction at the start. A BASIC program cannot have an input file and an output file open at the same time. If a file must be updated, the entire file must be read into memory, modified, and then written back to tape. Therefore, update processing is restricted to files which will fit entirely in memory. However, this restriction is not really too bad, since update processing must be done this way, if you only have one tape recorder connected to your AIM 65.

Approach

The AIM 65 itself creates text files by first invoking the monitor routine WHEREO. WHEREO establishes the tape recorder as the active output device, obtains a file name, and obtains a tape drive number (1 or 2). Each time a character is to be written to tape, AIM 65 will invoke the monitor routine OUTALL. If BASIC is going to write text files these same functions must be performed.

I have designed a machine language subroutine which allows BASIC to create text files. The text file output subroutine follows the convention established by out text input subroutine. The BASIC program must place the text line (or record) in the character string variable A$. Next, the BASIC program invokes the USR function. the machine language subroutine locates A$ in BASIC's memory and writes the contents of A$ to tape. After A$ has been written, control is given back to BASIC.

In this subroutine, I have used the capabilities of the USR function to pass data both to and from the machine language program. The argument of the USR function, (which BASIC passes to the machine language program) contains the number of bytes of data to be written from A$. In other words, you can set up A$ as an 80 byte string and write the first 25 bytes one time, the first 50 bytes the next time, and so on. Conversely, the machine language program passes data the other way—to BASIC. The value returned by the USR function indicates whether or not the write operation was successful.

AIM 65 users will note other similarities to our approach to reading text files. The AIM 65 will be put in the tape mode only for as long as it takes to write a record. Thus, the AIM 65 display is available as an output device between write operations. Data formatting considerations are simple: put what ever data you want into A$ and write it.

Loading the Subroutine

The hex dump of the machine code is shown in figure 1. In our system, the subroutine resides at location $7CA4. Since the subroutine is position independent, you may relocate it anywhere in memory without having to change a single byte of code.

If you prefer entering the code instruction format, the disassembly listing is included as figure 2. Just be careful of the absolute addresses which show up as operands of branch instructions.

Before testing out the subroutine, there is one address which your must check. It may vary from one version of the BASIC ROM to another. The machine language subroutine uses a BASIC subroutine to convert the USR argument from a floating point number to an integer. The address of this subroutine, not the subroutine itself, is contained in locations $B006 and $B007 of the BASIC ROM. Use the AIM 65 monitor to examine these locations. If they contain $FE and $BE respectively, then no changes are required. If they are different, however, you must modify the machine language subroutne. In this case, simply place the contents of location $B006 into location $7CF1, and place the contents of location $B007 into location $7CF2. All we are doing is telling the machine language subroutine where the BASIC floating point to integer conversion code is located.

Once your have loaded the subroutine and safely stored it on tape, you can initialize BASIC. Since the subroutine requires 148 bytes, you will have to account for this when responding the the MEMORY SIZE prompt. If you have a 4K system and you are only using the text file output subroutine, MEMORY SIZE would by 4096 minus 148 or 3948. If you are using both the text file input and output subroutines, MEMORY SIZE would by 4096 minus(148 + 164) or 3784.

Procedure

Now we're ready to go. The procedure for writing text files consists of the following four steps:

1. Open the file
2. Write a record
3. Test the return code
4. Close the file.

If you recall, out text file input subroutine closed the input file automatically. The text file output subroutine is different. It requires you to explicitly close the output file. This is necessary in order to make sure that the last block gets written to tape. We will illustrate these steps by going through a sample program. Our sample program will generate BASIC DATA statements. We will write these DATA statements to tape and then show how they can be appended to a BASIC program. this is one approach to saving and reusing data.

Step 1: Open the File

An output file is opened by POKEing location $F7 (247 decimal) to zero.

```
10 POKE 247,0
```

```
M>=7CA4 AD 13 A4 48
< > 7CA8 A5 75 85 F0
< > 7CAC A5 76 85 F1
< > 7CB0 A5 77 C5 F0
< > 7CB4 D0 12 A5 78
< > 7CB8 C5 F1 D0 0C
< > 7CBC A0 FF A2 FF
< > 7CC0 68 8D 13 A4
< > 7CC4 8A 6C 08 B0
< > 7CC8 A0 00 B1 F0
< > 7CCC C9 41 D0 07
< > 7CD0 C8 B1 F0 C9
< > 7CD4 80 F0 0D 18
< > 7CD8 A5 F0 69 07
< > 7CDC 85 F0 90 D0
< > 7CE0 E6 F1 D0 CC
< > 7CE4 A0 02 B1 F0
< > 7CE8 99 F0 00 C8
< > 7CEC C0 05 D0 F6
< > 7CF0 20 FE BE A5
< > 7CF4 AC D0 C5 A5
< > 7CF8 F7 D0 0A 20
< > 7CFC 71 E8 E6 F7
< > 7D00 AD 13 A4 85
< > 7D04 F8 A5 F8 8D
< > 7D08 13 A4 A5 F2
< > 7D0C C5 AD 90 AC
< > 7D10 A5 AD F0 17
< > 7D14 C9 51 B0 A4
< > 7D18 A0 00 B1 F3
< > 7D1C 20 BC E9 C8
< > 7D20 C4 AD D0 F6
< > 7D24 20 F0 E9 A2
< > 7D28 00 F0 95 20
< > 7D2C F0 E9 20 F0
< > 7D30 E9 20 0A E5
< > 7D34 A0 00 F0 EF
<
```

Figure 1: **Text File Output Subroutine — Hex Dump**

This will cause the machine language subroutine to invoke the AIM 65 monitor WHEREO. As we have seen, WHEREO will ask for the output device, file name and tape drive number.

Step 2: Write the Record

```
20 LN = 50000
30 FOR I = 1 TO 5
40  A$ = STR$(LN) + "DATA"
    + STR$(I)
50 POKE 4,103
60 POKE 5,125
70 Z = USR(LEN(A$))
```

Lines 20 and 30 are part of our sample application. Since we are generating DATA statements, we need to place line numbers in front of each one. Our generated line numbers start with 50000. Five DATA statements will be output. The text line is formatted in line 40. BASIC's STR$ function is used to convert numeric fields to character strings. The resultant line is placed in the character string A$. A$ is the output area. Each line of text to be written to tape must first be placed in A$. No other variable will do. Text data cannot be written to tape from any other variable without first being moved to A$.

Lines 50 and 60 tell BASIC where the machine language subroutine is located. The low order byte of the address (expressed in decimal) must be POKEd into location 4. Similarly, the high order byte of the address must be POKEd into location 5. In our example, the machine language subroutine is located at $7CA4. Be sure you adjust this for your particular configuration.

The USR function in line 70 causes the machine language subroutine to write the data from A$ to tape. Note that we've called the USR function with an argument. The argument tells the machine language subroutine how many bytes of A$ to write. If the argument was set to, say, five, then only the first five bytes of A$ would have been transferred to tape. by setting the argument to LEN(A$), we insure that the entire string will be written.

NOTE: in accordance with AIM 65 text file format, the machine language subroutine will automatically append a carriage return to each line of text written. You should not try to do this with BASIC. If you do, there will be two successive carriage returns on the tape—the subroutine's and yours. As far as the AIM 65 is concerned, this represents an end-of-file mark. When you go to read the tape, you won't be able to read very much of it.

Step 3: Test the Return Code

As line 70 shows, the USR function returns a value. This value is known appropriately as a return code. The return code can be assigned to any numeric variable (it doesn't have to be Z). The return code will tell you, from a software point of view, whether or not the write operation was successful. It won't tell you, for example, if your tape recorder is jammed or unplugged.

The return code can be interpreted as follows:

A: Return code is less than zero

If the return code has a value that is less than zero, then an error condition has been detected. There are four situations which will cause an error:

1. A$ is not defined
2. A$ is longer than 80 bytes
3. The USR argument is greater than 255
4. The USR argument is greater than LEN(A$)

Please note the 80 byte limit on the length of a text line.

B. Return code is greater than or equal to zero

If the value of the return code is greater than or equal to zero, then the machine language subroutine has successfully located A$ and has written its contents to tape. The return code will indicate the number of bytes written (exclusive of the carriage return).

Our sample program will test the return code like this:

```
80 IF Z    0 THEN STOP
90 LN = LN + 10
100 NEXT I
```

***Figure 2:* Text File Output-Subroutine—Instruction Format**

```
K>*=7CA4
/39
7CA4 AD LDA A413        Save OUTFLG on the stack
7CA7 48 PHA
7CA8 A5 LDA 75          Start of BASIC's symbol table
7CAA 85 STA F0
7CAC A5 LDA 76
7CAE 85 STA F1
7CB0 A5 LDA 77          Is it the end of the symbol table?
7CB2 C5 CMP F0
7CB4 D0 BNE 7CC8        No...
7CB6 A5 LDA 78
7CB8 C5 CMP F1
7CBA D0 BNE 7CC8        No...
7CBC A0 LDY #FF         Error exit. Set return code to -1
7CBE A2 LDX #FF
7CC0 68 PLA             Normal exit. Restore OUTFLG
7CC1 8D STA A413
7CC4 8A TXA
7CC5 6C JMP (B008)      Back to BASIC
7CC8 A0 LDY #00
7CCA B1 LDA (F0),Y
7CCC C9 CMP #41         Is it A$?
7CCE D0 BNE 7CD7
7CD0 C8 INY
7CD1 B1 LDA (F0),Y
7CD3 C9 CMP #80
7CD5 F0 BEQ 7CE4
7CD7 18 CLC             Set up for next symbol table entry
7CD8 A5 LDA F0
7CDA 69 ADC #07
7CDC 85 STA F0
7CDE 90 BCC 7CB0
7CE0 E6 INC F1
7CE2 D0 BNE 7CB0
7CE4 A0 LDY #02         Get address and length of A$
7CE6 B1 LDA (F0),Y
7CE8 99 STA 00F0,Y
7CEB C8 INY
7CEC C0 CPY #05
7CEE D0 BNE 7CE6
7CF0 20 JSR BEFE        Convert USR argument to integer
7CF3 A5 LDA AC          Is it greater than 255?
7CF5 D0 BNE 7CBC        Yes, then error
7CF7 A5 LDA F7          First time through?
7CF9 D0 BNE 7D05        No...
7CFB 20 JSR E871        Yes, call WHEREO
7CFE E6 INC F7
7D00 AD LDA A413        Pick up new OUTFLG and
7D03 85 STA F8          Save it in a temporary variable
7D05 A5 LDA F8          Restore OUTFLG from the temporary
```

Figure 2 (continued)

```
7D07 8D STA A413
7D0A A5 LDA F2
7D0C C5 CMP AD
7D0E 90 BCC 7CBC      Error if USR arg greater than LEN(A$)
7D10 A5 LDA AD
7D12 F0 BEQ 7D2B      Caller says it's time to close the file
7D14 C9 CMP #51
7D16 B0 BCS 7CBC      Error if USR arg greater than 80
7D18 A0 LDY #00
7D1A B1 LDA (F3),Y    Pick up a byte from A$
7D1C 20 JSR E9BC      Use OUTALL to write it
7D1F C8 INY
7D20 C4 CPY AD
7D22 D0 BNE 7D1A
7D24 20 JSR E9F0      CRLF marks the end of the line
7D27 A2 LDX #00
7D29 F0 BEQ 7CC0      Exit
7D2B 20 JSR E9F0      Close the file. Two CRLFs
7D2E 20 JSR E9F0
7D31 20 JSR E50A      DU11 writes the last block
7D34 A0 LDY #00       Set return code to 0
7D36 F0 BEQ 7D27      Exit
```

So, if there is some kind of error, the program will terminate with a BREAK message.

Lines 90 and 100 set up the next DATA statement line number and finish up the loop.

Step 4: Close the File

When we have finished writing all the records we want, we must "close" the output file. There are several actions that must be done in order to close a file. First of all, it we are writing a text file that contains BASIC source program statements, we must write a control-z at the end of the file. (Refer to the *Basic Reference Manual,* page G-3). In any case, the text file must be terminated with two carriage returns. Our machine language subroutine will take care of writing the two carriage returns. However, since the machine language subroutine has no idea of whether the text file that we are writing is a BASIC source program or not, we must write the control-z ourselves.

In our sample program, the code to close the output file is:

```
110  A$=CHR$(26):  REM
  CONTROL Z
120 Z=USR(1)
130  Z=USR(0):  REM  CLOSE
  FILE
END
```

Lines 110 and 120 write a control-z at the end of the text file.

Calling the machine language subroutine with the argument of the USR function set to zero, closes the text file. The machine language subroutine will output two consecutive carriage returns. Next, it will write the last block of data from the AIM 65 output buffer to tape. Lastly, it will turn both tape recorders (drives 1 and 2) on.

Sample Program

Figure 3 shows a complete listng of the sample program and a test run. You should be able to duplicate the results exactly.

The sample program generates five DATA statements. These are written to a tape file. Next, the tape is read with the BASIC LOAD command (without first typing NEW). A LIST of the program reveals that not only was the tape write succesful but also that the DATA statements were appended to our sample program. Please recognize that this is a sample program. We generated DATA statements only for the sake of simplicity. There is no reason why we could'nt have created and written to tape an entire BASIC program.

We have described a machine language subroutine which opens up the capability to create text files from BASIC. You can use this capability for any number of applications. Just keep in mind the restriction that was mentioned earlier in the article: an input file and an output file cannot be open at the same time in the same program.

Subroutine Logic

Figure 4 contains the Warnier-Orr diagram of the machine language subroutine. With this diagram and the description that follows, you should be able to modify the subroutine to fit your particular needs.(To broadly review Warnier-Orr diagrams, the sequence of operations is determined by reading from the top of the diagram to the bottom. Hierarchy is indicated by reading from left to right).

A description of the zero page variable used in the subroutine is included as figure 5. If you are using our text file input subroutine, you will notice that many of the same zero page locations are used. There is no real conflict, however. Both the text file input and output subroutine initialize locations $F0 through $F4 each time they are called.

Upon entry to the text file output subroutine, the AIM 65 variable OUTFLG is saved on the stack. This allows us to preserve the AIM 65 active output device indicator between subroutine calls. In other words, assuming that the display/printer is the active output device, it will be disabled while the subroutine is using the tape recorder as the active output device. Next, one of two lower level routines is invoked, depending on whether or not A$ has been defined by the BASIC program. When control is again received from one of these lower level routines, OUTFLG will be pulled from the stack. This restores the original active output device (for example the display/printer). Finally, the

Figure 3: Sample Program

STEP 1

Key in and LIST the sample program. WARNING: The subroutine is located at $7CA4 as specified by lines 50 and 60. You may need to change this for your system.

```
10 POKE 247,0
20 LN=50000
30 FOR I=1 TO 5
40 A$=STR$(LN)+"DATA"+STR$(I)
50 POKE 4,103
60 POKE 5,125
70 Z=USR(LEN(A$))
80 IF Z<0 THEN STOP
90 LN=LN+10
100 NEXT I
110 A$=CHR$(26):REM CONTROL-Z
120 Z=USR(1)
130 Z=USR(0):REM CLOSE FILE
140 END
```

STEP 2

RUN then program. It will write 1 block of data to tape TEST1.

```
RUN
OUT=T F=TEST1 T=2
00
```

STEP 3

LOAD tape TEST1 (do not type NEW). 1 block of data will be read. The data will be displayed as it is processed.

```
LOAD
IN=T F=TEST1 T=1
00 SRCH F=TEST1 BLK= 00 LOAD
 50000DATA 1
 50010DATA 2
 50020DATA 3
 50030DATA 4
 50040DATA 5
```

STEP 4

LIST the program. The generated DATA have been appended to the original program.

```
10 POKE 247,0
20 LN=50000
30 FOR I=1 TO 5
40 A$=STR$(LN)+"DATA"+STR$(I)
50 POKE 4,103
60 POKE 5,125
70 Z=USR(LEN(A$))
80 IF Z<0 THEN STOP
90 LN=LN+10
100 NEXT I
110 A$=CHR$(26):REM CONTROL-Z
120 Z=USR(1)
130 Z=USR(0):REM CLOSE FILE
140 END
50000 DATA 1
50010 DATA 2
50020 DATA 3
50030 DATA 4
50040 DATA 5
```

machine language subroutine returns to BASIC. This is done by issuing a JMP indirect to location $B008 in the BASIC ROM. $B008 converts the 16-bit return code (stored in A and Y) to a floating point number.

If A$ is defined, a call will be made to a subroutine in the BASIC ROM. This subroutine converts the argument of the USR function to a 16-bit integer. (Refer to page F-1 of the *BASIC Reference Manual.)* The value of the 16-bit integer is examined and one of two lower level routines is invoked as appropriate.

If A$ is not defined, then no output record exists. This is probably an error. The machine language subroutine sets the return code to −1 to signal the error condition.

In the event that the argument of the USR function is 255 or less, the following steps will be carried out. First, if the machine language subroutine is being called for the first time, lower level initialization code will be invoked. In any case, OUTFLG is restored from the temporary variable located at $F8. Normally, this will put the AIM 65 in the tape mode. Then, the USR argument (that is, the number of bytes to be written) is compared with the actual length of A$.

Should the USR argument specify a value greater than 255, an error condition exists. Microsoft BASIC does not permit strings longer than 255 characters. Therefore, the machine language subroutine sets the return code to *−1.*

If the machine language subroutine is being called for the first time, WHEREO will be called. This AIM 65 monitor subroutine will prompt the user for the output device, file name, and tape drive number. WHEREO also sets OUTFLG with a new value. We store the new value in OUTFLG in the temporary variable at $F8.

If the USR argument is less than or equal to the length of A$, then processing can continue. We test the USR argument for three conditions:

A. USR argument is 0
B. USR argument is non-zero and less than or equal to 80

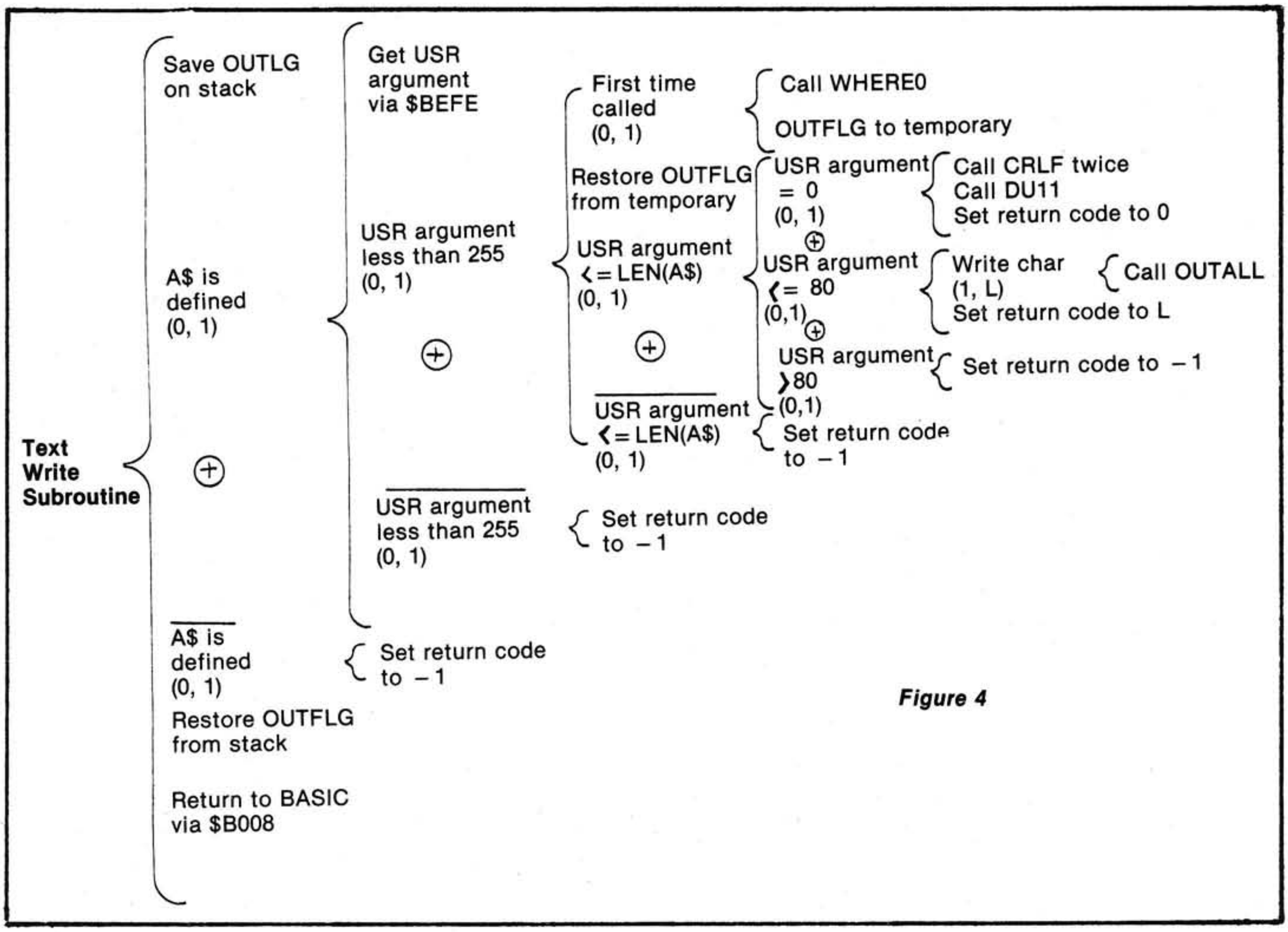


Figure 4

C. USR argument is greater than 80.

If, on the other hand, the USR argument is greater than the length of A$, there is some inconsistency. The machine language subroutine is being asked to write more data than is actually present. So, in this case, an error condition is raised and the return code is set to −1.

An output file is closed by setting the USR argument to zero (condition A above). The following actions take place. The AIM 65 monitor routine CRLF is called twice. This puts two successive carriage returns on the tape as an end-of-file mark. Next, the monitor routine DU11 is called. DU11 writes the last tape block and turns on *both* tape drives. Finally, we set the return code to zero and exit.

If the USR argument is greater than zero and less than or equal to 80, we output the number of bytes specified by the USR argument. The AIM 65 subroutine OUTALL performs the output operation. The end of the text line is marked by calling CRLF. The return code is the set to the number of bytes written (exclusive of the carriage return).

If the USR argurment is greater than 80, the return code is set to −1 to indicate an error. This is because we have established a maximum record length or 80 bytes for our text file input and output operations. This limitation is easily relaxed, however.

μ

***Figure 5:* Zero Page Variables**

SYMTAB	$F0, $F1	Pointer to BASIC's symbol table
LEN	$F2	Length of A$
APNT	$F3, $F4	Pointer of A$ in BASIC's memory
TEMP2	$F7	First time switch
TOTFLG	$F8	OUTFLG hold area

$$ WIN $$

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

The MICRO Software Catalog: XXVI

Software announcements for the 6502 based systems

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

Name: **Electric Bill**
System: ATARI OR PET
Memory: 8K
Language: BASIC
Hardware: no extra
Description: A household utility program designed to predict what your next month's electric bill is going to be BEFORE you use the electicity. By using this program weekly, you can watch your consumption rate, cutting excessive electrical use. Software package comes with BASIC program on cassette and six pages of documentation.
Price: $4.95 for PET version
$5.95 for ATARI version
Author: **Mitchell Kelly**
Advanced Data Services
Available: **Kinetic Designs**
401 Monument Rd.#123
Jacksonville, FL 32211

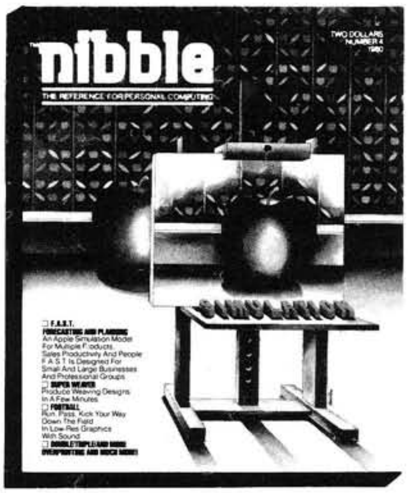
Name: **CRS/PDOS**
System: PET-Old or New Roms
Memory: 16K, 24K, or 32K
Language: Machine Language
Description: A comprehensive disk operating system for the Microtech PEDISK — floppy disk system for PET. The system was designed to be easy to use, simple syntax, yet still provide powerful data file handling capabilities. It greatly facilitates the use of languages other than BASIC and a conditional assembler/editor is available. Its use does not interfere with any feature of the PET. All commands and keywords function as they do in a PET without CRS/PDOS. Disk functions are provided by new commands added to PET BASIC for disk program and data file control. Can be modified.
Price: $75.00 (available with EXS 100 board)
Available: **CGRS Microtech, Inc.**
P.O. Box 102
Langhorne, PA 19047

Name: **GROCERY LIST**
System: APPLE II or APPLE II +
Memory: 48K
Language: ROM or RAM Applesoft
Hardware: DISK II and PRINTER
Description: An easy-to-use, self prompting program that lets the user scan a Master File of up to 500 grocery and related items. With a single key-stroke the shopper may select the quantity and item dsired. Prints a shopping list organized by table number, section number, or department, such as "DARY", "DELI", or any other four letter code. The code is user defined. Over 150 items on a Master File are included. A Master File Editor is also included.
Price: $19.95 on disk
Includes: Diskette and complete documentation
Author: **D.R. Jarvis**
Available: **D. R. JARVIS COMPUTING**
1039 Cadiz Dr.
Simi, CA 93065

Name: **C1 Shorthand**
System: OSI C1 (and Superboard) BASIC-in-ROM
Memory: Any size system
Language: Basic and machine
Hardware: None special
Description: Shorthand is a BASIC program that POKES in a machine language routine that will allow you to use two keys to enter BASIC keywords and commands. Saves much time in typing in long programs. C2, C4, and C8 versions are about to be released. Write for availability.
Copies: Just released
Price: $12 on cassette tape, ppd.
Author: **Mike Cohen**
Available: **Orion Software Associates**
147 Main Street
Ossining, N.Y. 10562

Name: **Magic Boot**
System: Ohio Scientific
Hardware: Disk
Software: 0S65D
Memory: Uses 2K
Description: Adds renumber, full cursor editing, memory pack, screen clear and color background to BASIC. All routines are superfast machine code and handle All cases. These are not USR (x) routines, but extensions to BASIC itself. All you do is boot and these commands are ready to use—in immediate mode or in your programs. Also includes new BEXEC* and disk utilities. Write for more information or separate price list.
Price: $50.00
Available: **Universal Systems**
1647 E. Old Shakopee Rd.
Minneapolis, MN 55420

"NIBBLE® IS TERRIFIC" (For Your Apple)



NIBBLE IS: *The Reference for Apple computing!*

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

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

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

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

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

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

In coming issues, look for:

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

And many many more!

NIBBLE is focused completely on the Apple Computer systems.

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

Try a NIBBLE!

nibble No. 4

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

I'll try nibble!

Enclosed is my $15 (for one year).

☐ **check** ☐ **money order**

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

Name ______________________

Address ______________________

City ______________________

State ______________ Zip ________

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

6502 Bibliography: Part XXVI

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

721. Washington Apple Pi 2, No. 1 (Jan. 1980)

Mitchell, Howie, "Master Catalog Program," pg. 7.
A catalog program which works well for finding diskettes containing specific programs.

722. The Harvest 1 No. 8 (January 1980)

Pfeiffer, Jim, "How Applesoft Stores String Arrays and a Tip on How to Clear String Array Space," pgs. 1-7.
A tutorial for the APPLE.

Anon., "APPLE Configuration," pg. 6.
This program figures out which cards your APPLE has in which peripheral slot.

Neiburger, Skip, "Label Maker," pg. 9.
Print labels with a maximum of 4 lines and 28 characters in length.

723. The Paper 3 Issues 2/3 (March/April 1980)

Busdiecker, Roy, "The End," pgs. 19-21.
A fast routine to find the end of whatever BASIC program that happens to be in your PET.

Barroll, Ken C., "A Simple Mail Label Program," pg. 23.
Short Basic program for your printer.

Sparks, Paul W., "A Screen Print Machine Language Program, " pgs. 24-33.
A cookbook tutorial including assembly language programming.

724. APPLE Cookbook 1, No. 1 (January 1980)

Matsumoto, Tom, "Pascal Turtle Graphics," pgs. 2-3.
How to control the Turtle head on the APPLE/Pascal system.

725. APPLE Cookbook 1, No. 1 (January 1980)

Wigginton, R. "Saving Character Strings on Cassette Tape," pg. 8.
If you need to save strings to tape, this program will do very well on your Apple.

Golding, Val and Aldrich, Darrell, "Two Short Utilities," pg. 11.
"Disk Space" and "C for Catalog" are listed for the APPLE.

726. APPLE Peel 2, No. 2 (February 1980)

Brown, Tom, "A New Dimension in Applesoft Programming," pgs. 8-10.
Redimension your Applesoft array to a larger or smaller size.

727. The Harvest 1, No. 9 (February 1980)

Sander-Cederlof, Bob, "S-C Assembler Auto Line Number," pgs. 4-6.
Add Auto-Number to your S-C Assembler II.

728. Washington APPLE Pi 2, No. 2 (February 1980)

Field, Bruce F., "Remote Temperature Measurement with the APPLE," pgs. 16-17.
Hook up a thermistor sensor to your APPLE.

729. APPLE Cookbook 1, No. 2 (March/April 1980)

Matsumoto, Tom, "Pascal Tutor," pgs. 2-4.
The second in this tutorial series on Pascal for the APPLE.

730. Rainbow 2, Issues 3/4 (March/April 1980)

Lee, Montgomery, "Printing Text on Page 2," pgs. 6-7.
Hints for the APPLE.

Frazer, Fred, "The Search Goes On," pgs. 34-35.
A faster version of a program to search for Pythagorean Triplets, on the APPLE.

731. The Harvest 1, No A (March 1980)

Russ, John C., "Simulation with the APPLE, or some thoughts on RND (NUMBERS)," pgs. 1-4.
Several examples of the use of random numbers in listings.

732. APPLE Peel 2, No. 3 (March 1980)

Little, Gary, "EDIT + : Who needs Autostart ROM?" pg. 3.
A program to provide editing features such as those offered on the Autostart ROM of the APPLE.

733. From the Core (April 1980)

Anderson, Chip, "The Processor," pg. 11.
A utility program for the APPLE.

734. Stems from APPLE 3, No. 4 (April 1980)

Ward, Dennis, "Applesoft Debugger," pg. 7.
Routine to print error codes in debugging APPLE programs.

Jochumson, Chris, "APPLE HI-RES Graphics," pgs. 8-11.
A tutorial for Hires on the APPLE.

Reed, Ron, "APPLE Language System Pascal Patch," pgs. 16-17.
How to live with the unruly cursor. Eliminates the flashing upper case.

735. The Harvest 1, No. B (April 1980)

Hobson, Blake, "DEF FN Defined," pgs. 6-7.

A program is given in which the DEF FN statement is changed with each input, thus results in a program which modifies itself.

736. O.S.I. Users Independent Newsletter 3 (May 1980)

Curley, Charles, "OS-65D Addresses," pg. 2.
A list of useful addresses for the OS-65D with comments and observations.

737. Kilobaud Microcomputing No 41 (May 1980)

Baker, Robert W., "PET-Pourri," pgs. 7-8.
Comments on Tape drive with a counter, programming tips, software and workbooks.

Garrett, Robert D., "Machine Language Routine for Challenger II," pg. 18.
How to move a machine language program.

Williams, John D., "Software for the AIM 65," pgs. 96-98.
Discussion of some of the mysteries of the AIM 65 monitor subroutines.

Hoyt, Bruce, "Keeping Real Time with OSI's Superboard II," pgs. 202-203.
A real-time clock, with alarm and timer, for the 6502.

738. Washington Apple Pi 2, No. 4 (April 1980)

Pilloff, Hersch, "An Error in DOS 3.2.1," pg. 7.
Under certain conditions the APPEND function will overwrite the previous sectors in the text file. Here is a fix.

Crossman, Craig, "APPLE Tricks," pg. 11.
Learn how to INIT a disk in half the time, type 'unaccessable' characters, make your program unlistable, etc. in this article on the APPLE.

739. Cider Press (April 1980)

Yee, David R., "Stop the Blinking Cursor," pg. 8.
Routine to alter the cursor on the Apple.

Slater, Michael, "Apple Pascal Compiler," pg. 9.
How to eliminate a bug in the compiler.

Anon., "APPEND FIX, DOS 3.2.1," pg. 10.
Using this method, one never worries about APPEND overwriting the start of a file on the Apple Disk.

Uhley, John, "Integer Basic Super-Lock," pg. 11.
A Machine Language program which alters an Integer Basic program using a code word which the user enters.

740. Creative Computing 6, No. 5 (May, 1980)

Geiger, Rick, "Apple Strings," pg. 118-122.
Discussion of the use of strings on the Apple.

Carpenter, Chuck, "Apple-Cart," pg. 158-165.
Comparison of Apple II vs. Apple II Plus, Converting Integer Basic to Applesoft, Assembly Language, Monitor Commands, OpCodes, Address Modes, etc.

741. Byte 5, No. 5 (May, 1980)

Helmers, Carl, "Computer-Controlled Viewing of the 1980 Eclipse," pg. 6.
Use of an Apple II Pascal system controlling the photographing of the eclipse.

Swank, Joel, "KIMDOS," pg. 44-50.
Using your KIM-1 with a Percom Floppy-Disk Drive.

Ezard, Lawrence A., "Formatted Program Output for the KIM-1," Pg. 190-194.
Here is a short KIM-1 program to find bugs, print out and document programs, etc.

742. Interface Age 5, Issue 6 (June, 1980)

Mallon, Marvin, "Teach Your PET to Read 'Marked Cards'," pg. 56-61.
With your PET and a card reader you can enter data with the IBM cards.

Fox, Tom, "Looking at Micro-Based Business Systems," pg. 70-78.
6502 Based Micros are discussed in this article.

Wetzel, Ken, "A Break Service Routine for a KIM-1 with a Teletype," pg. 132-135.
The use of software interrupts, or breakpoints, are recognized debugging aids. This listing for the KIM lists the contents of the registers on the interrupt.

743. MICRO Issue 24 (May, 1980)

DePriest, Dale, "Plotting with Special Character Graphics," pg. 11-13.
A primer on generating plot mode type graphics with special characters, with special listings for PET and Challenger.

Wells, George, "SYM-1 Basic 'GET' Command," pg. 15-17.
The use of the 'GET' function is discussed and several examples for the SYM-1 are provided.

DeJong, Marvin L., "A Simple Temperature Measurement Program and Interface," pg. 19-23.
Using a micro for temperature measurement demonstrates problems involved in interfacing the real world.

Wevers, Henk J., "Shorthand Commands for Superboard II and Challenger CIP Basics," pg. 25-27.
How to intercept the BASIC's input routine and how to implement a shorthand notation.

Wilson, W.E., "A Formatted Dump Routine for the AIM 65," pg. 29-30.
Control the formatting of the dump to conform with the user's printer capabilities with this routine.

Tulloch, Michael, "New and Better PET User Port Printer Routines," pg. 33-37.
A series of programs are presented which drive any TTL, parallel, or ASCII printer from the PET's user port.

Taylor, William L., "Graphics and the Challenger CIP, Part 5," pg. 41-43.
The final installment in the series discusses plotting techniques and moving characters.

Orton, Ralph R., "SYM-1 Sends Morse Code," pg. 49-51.
Use your SYM as a Morse Code teaching tool, automatic ID'er or 'canned' message sender.

Reynolds, Lee, "An EDIT Mask Routine in Applesoft BASIC," pg. 53-57.
These routines permit you to produce professional looking output on your Apple.

Tenny, Ralph, "Expand KIM-1 Versatility in Systems Applications," pg. 63-64.
Techniques and programs permitting the simple addition of six sense switches or an ASCII keyboard to the KIM.

744. Recreational Computing 8, No. 6, Issue 45 (May/June 1980)

Gollux, Ernest, "Programming the 6502 in Machine Language," pgs. 12-15.
A tutorial on machine language for 6502 micros.

Trenholme, John, "Pronounceable Names," pg. 21.
A name generator for the PET.

Trenholme, John, "PET Input without Breaks," pg. 21.
Useful routine for the PET.

Opedal, Donald R., "Editing Softape Prefix Programs," pg. 36.
How to manipulate those Softape programs.

Advertisers' Index

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(TM). 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(TM), Applesoft II(TM)

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(TM), Applesoft II(TM)

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(TM), Applesoft II(TM).

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(TM), Applesoft II(TM).

* Apple, Apple II Plus, Disk II and APPLESOFT II are trademarks of Apple Computer Company.
* Micromodem II is a trademark of D.C. Hayes Associates, Inc.

The home computer you thought was years away is here.

C8P DF

Ohio Scientific's top of the line personal computer, the C8P DF. This system incorporates the most advanced technology now available in standard configurations and add-on options. The C8P DF has full capabilities as a personal computer, a small business computer, a home monitoring security system and an advanced process controller.

Personal Computer Features
The C8P DF features ultra-fast program execution. The standard model is twice as fast as other personal computers such as the Apple II and PET. The computer system is available with a GT option which nearly doubles the speed again, making it comparable to high end mini-computer systems. High speed execution makes elaborate video animation possible as well as other I/O functions which until now, have not been possible. The C8P DF features Ohio Scientific's 32 x 64 character display with graphics and gaming elements for an effective resolution of 256 x 512 points and up to 16 colors. Other features for personal use include a programmable tone generator from 200 to 20KHz and an 8 bit companding digital to analog converter for music and voice output, 2-8 axis joystick interfaces, and 2-10 key pad interfaces. Hundreds of personal applications, games and educational software packages are currently available for use with the C8P DF.

Business Applications
The C8P DF utilizes full size 8″ floppy disks and is compatible with Ohio Scientific's advanced small business operating system, OS-65U and two types of information management systems, OS-MDMS and OS-DMS.
The computer system comes standard with a high-speed printer interface and a modem interface. It features a full 53-key ASCII keyboard as well as 2048 character display with upper and lower case for business and word processing applications.

Home Control
The C8P DF has the most advanced home monitoring and control capabilities ever offered in a computer system. It incorporates a real time clock and a unique FOREGROUND/BACKGROUND operating system which allows the computer to function with normal BASIC programs at the same time it is monitoring external devices. The C8P DF comes standard with an AC remote control interface which allows it to control a wide range of AC appliances and lights remotely without wiring and an interface for home security systems which monitors fire, intrusion, car theft, water levels and freezer temperature, all without messy wiring. In addition, the C8P DF can accept Ohio Scientific's Votrax voice I/O board and/or Ohio Scientific's new universal telephone interface (UTI). The telephone interface connects the computer to any touch-tone or rotary dial telephone line. The computer system is able to answer calls, initiate calls and communicate via touch-tone signals, voice output or 300 baud modem signals. It can accept and decode touch-tone signals, 300 baud modem signals and record incoming voice messages.

These features collectively give the C8P DF capabilities to monitor and control home functions with almost human-like capabilities.

Process Controller
The C8P DF incorporates a real time clock, FOREGROUND/BACKGROUND operation and 16 parallel I/O lines. Additionally a universal accessory BUS connector is accessible at the back of the computer to plug in additional 48 lines of parallel I/O and/or a complete analog signal I/O board with A/D and D/A and multiplexers.

Clearly, the C8P DF beats all existing small computers in conventional specifications plus it has capabilities far beyond any other computer system on the market today.

C8P DF is an 8-slot mainframe class computer with 32K static RAM, dual 8″ floppies, and several open slots for expansion.

Prices start at under $3,000.

Computers come with keyboards and floppies where specified. Other equipment shown is optional.

For literature and the name of your local dealer, CALL 1-800-321-6850 TOLL FREE.

OHIO SCIENTIFIC
1333 SOUTH CHILLICOTHE ROAD
AURORA, OH 44202 • [216] 831-5600