MICRO™

The Magazine of the APPLE, KIM, PET

and Other 6502 Systems

SPELUNKER

BUT ONLY FOR THE BRAVE !

NO 17 OCTOBER 1979 $2.00

NOW PRESENTING...

Apple® software

for your Entertainment · Business · Education

Star Attractions:

FILEMASTER 2 programs: *FORMAT & RETRIEVAL* comprise a powerful data file manager. Great for everything from phone lists to legal abstracts. Needs 32K. Design your own data structure. Up to 500 characters per record. Up to 15 searchable fields in any combination. **On Disk** **$34.95**

SPACE Multi-faceted simulation of life in interstellar society. You and opponents must make life & death decisions. Keeps track of your progress from one game to next. Needs 48K and Applesoft ROM. **Disk** **$29.95**

Pot O'Gold I or our All New Pot O' Gold II A collection of 49 programs for 16K Apple. Everything from Logic to action games. Only a buck a game. Specify I or II. Price each: **Tape $49** . . . . **Disk $54**

ADVENTURE Fight off pirates and vicious dwarfs. 700 travel options, 140 locations, 64 objects. Needs ROM & 48K. **Disk.** . **$29.95**

16K CASSETTE INVENTORY Use item number, description, stock amount, reorder amount, restock date, cost & sell price. Holds up to 140 items. **Tape** **$35**

32K DISK INVENTORY: Use stock numbers description, vendor, record of purchase and sales date, amount on hand, cost & sell price, total value. Holds up to 300 items. **Disk** **$40**
With Parts Explosion: Disk **$50**

32K DATA BASE Cross file for phone lists, bibliographies, recipes. Run up to 9 lines of 40 columns each. Search by item anywhere. **Disk.** **$20**

24K HI-RES LIFE SIMULATION Conway's equations on 296x180 screen. A mathematical simulation to demo population growth with birth, death and survival as factors. **Tape** **$10**

16K CIRCUIT LOGIC DEVELOPMENT AID Evaluate circuits of up to 255 gates, including AND, OR, NOR, NAND, XOR, XNOR and INVERTER. **Tape** **$10**

16K MORSE CODE TRAINER Learn Morse Code, and transmit or receive over radio. **Tape** **$10**

16K DEVIL'S DUNGEON: Adventure through dark passages where monsters, demons, poisonous gas, dropoffs threaten . . . all to discover fantastic treasures. Comes with instruction book. **Tape . . . $10**

16K PACIFICA: Discover the floating island and rescue the beautiful princess. To win you must recover the enchanted crown, but you face the threat of magic spells and demons. **Tape** **$9.95**

Don't see what you've been looking for, here? Then write for our FREE SOFTWARE CATALOG. We're saving one just for you!

To order software, add $2 shipping. To transfer tape versions to disk add $5. California residents add 6% sales tax. Sorry, we can not ship to P. O. Boxes. VISA/MASTERCHARGE Welcomed!

RAINBOW'S CASINO 9 gambling games: Roulette, Blackjack, Craps, Horserace, and a few originals that Vegas hasn't heard about. Needs 16K. **Tape** **$29.95**

16K SPACE WAR: You in your space capsule battle against the computer's saucer . . . in hi-res graphics. **Tape** **$12**

16K MEMORY VERIFY Diagnostic routine to check range of memory. Indicates faulty addresses, data in memory cell, and faulty data. **Tape** **$5**

16K APPLEODION Music synthesis composes original Irish jigs. Enter your own music and save on tape or disk. Includes 3 Bach fugues. **Tape.** **$10**

16K APPLEVISION Demo for Hi-Res graphics and music. **Tape** **$10**

32K COMPU-READ 5 programs to teach you speed reading, in stages. Includes synonym and antonym identification. You control your rate of speed, or keep up with the computer's pace. **Disk.** **$24.95**

48K PERCEPTION I, II, III random shapes and sizes must be matched. In III, you control format and display time and get weighted scores. Needs ROM. **Each Disk.** **$24.95**

32K STORY TELLER Use your bizarre imagination and input key words for fantastic and funny tales. Never the same story twice. **Tape** **$12.95**

32K WAR/RESCUE Engage in 10 battles with your infantry against the Apple robots. Calculate Apple's strategy and win more battles than the computer. **Tape.** **$12.95**

24K POLAR PLOT Plot polar equations in Hi-Res Graphics. **Tape** **$10**

32K SHAPE SCALER Utility to generate and animate Hi-Res graphic shapes. Simple routine provided to inspect position of shapes, and specify precise X/Y coordinates and scale. Needs ROM. **Disk.** **$13.95**

32K ZINTAR/PROPHET Great party game. Under control of the mighty Zintar's edict you take a very special trip to the world of Krintar. Heightened visual graphics. Needs ROM. **Disk** **$16.95**

APPLE MONITOR PEELED Everything you wanted to know about the Apple Monitor but couldn't figure out. User-written manual in plain English clears your confusion. **Only** **$9.95**

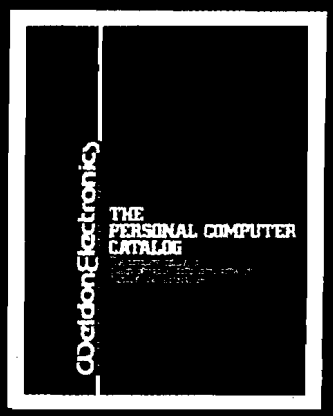
RAINBOW COMPUTING INC.

Garden Plaza Shopping Center, Dept. 11A
9719 Reseda Blvd., Northridge, Ca 91324
Telephone: (213) 349-5560

RUN THIS PROGRAM

10 Enter data in form below
20 Goto mailbox
30 Mail form
40 Receive the Personal
Computer Catalog
50 End

Well Done!



Follow this simple program and you will receive The Personal Computer Catalog. The one reference book to fine quality personal computers, software, supplies and accessories.

This valuable catalog is FREE so mail your order today.

Name _____

Address _____

City _____ State _____ Zip _____

Do you own a computer? _____ What type? _____

Do you use your computer for: Business? _____

Personal? _____ Education? _____ Other? _____

Mail this form to:

WeldonElectronics

SERVING THE PERSONAL COMPUTER INDUSTRY

Or phone: (612) 884-1475

Weldon Electronics
4150 Hillcrest Road
Wayzata, MN 55391

The Basic Switch™

Attention "Old" Pet™ Owners:

Not sure about the ROM Retrofit Kit from Commodore?
Now you can use **both** sets of Commodore ROMs and others as well.

The Basic Switch allows switch selection of **either** ROM set (your original set or your retrofit set) from Commodore. Plus, Model 15-A includes an additional zero insertion force socket allowing easy use of ROMs like the BASIC Programmer's Toolkit ... concurrently.

Model 14-E The economy model of **The Basic Switch.** Stand alone board and harness without case and case hardware. The free standing unit is ready to accept your ROMs.

Model 14-D Same as Model 14-E but includes attractive protective case and mounted **Basic Switch** board.

Note that Model 14 Series does not allow for expansion ROMs like the BASIC Programmer's Toolkit.

Model 15-A **The Basic Switch** plus ... includes expanded cable assembly and zero insertion force socket. Your 15th ROM simply plugs in ... enabled while either ROM set is selected. Socket 15 may be readdressed by the user for additional flexibility.

The Basic Switch is sold in assembled form only. All models are designed for easy attachment to your Pet with a convenient cable assembly. No soldering or drilling is required. **The Basic Switch** mates with a cable assembly at your primary board, and **does not use** the physical connectors of any Pet ports.

Model 15-A allows you to use the BASIC Programmer's Toolkit without the need for the additional $25.00 board or tying up your ports. And since we've designed the 15th socket to be readdressable, watch for more ROM pacs later in the Fall.

The Basic Switch:		With **installed** ROM Retrofit Kit from Commodore:	With BASIC Programmer's Toolkit *
Model 14-E	$64.95	$149.95	–
Model 14-D	$77.95	$162.95	–
Model 15-A	$99.95	$184.95	$149.95

Model 15-A with installed ROM Retrofit and Basic Programmer's Toolkit: $229.95

Model 15-A with installed ROM Retrofit and both Toolkits: $274.95

"Old" Pets were shipped with 24 or 28 pin ROMs. You must check which you have, and specify at time of order.

The Basic Switch™ is a product of

Small System Services, Inc.
900 Spring Garden Street
Greensboro, North Carolina 27403
Telephone 919-272-4867

North Carolina residents add 4% sales tax. All orders add $2.50 shipping.

MICRO™

Table of Contents

October 1979
Issue Number 17

Staff

Editor/Publisher
Robert M. Tripp

Assistant Editor
Mary Ann Curtis

Business Manager
Maggie E. Fisher

Circulation Manager
Carol Ann Stark

Production Assistant
L. Catherine Bland

Comptroller
Donna M. Tripp

Advertiser's Index

SUPER-TEXT is a professional word processing system for the Apple II and Apple II Plus computers.
SUPER-TEXT is the most innovative word processor available on any personal microcomputer and includes features previously found only on word processing systems costing thousands more!
An advanced multiple paging system allows you to view two text screens simultaneously. You may keep notes or instructions on one text screen while you edit on the other.
SUPER-TEXT is a character oriented editor with complete cursor controls to easily move the cursor to any position in the text with a minimum of keystrokes.
Built in floating point math and automatic tabbing facilitate the preparation of all manual reports including financial reports, insurance forms, real estate settlements and more.
SUPER-TEXT is easier to operate than a typewriter yet challenges the flexibility of pencil and paper.

SELECTED FEATURES:

EDITING - Full floating cursor. Cursor control - left, right, up, down or to center of screen. Add or insert a character, word or line. Automatic carriage return eliminates a word breaking at the end of the screen line. Delete a character, word, line or screen. Automatic on screen tabbing and right or left justification of columns. Unlimited text movement. Scroll either a page or a line forward or back. Move to the beginning or end of the text or screen. Move to the last change made in the text. Move to a block marker. Global search and replace (selective or all). Block operations - copy, delete or save to disk. Select multiple or single screen mode.
ADVANCED FILE HANDLING - Requires only two keystrokes to load or save a file to disk. The file name does not have to be entered which eliminates misspelling and "FILE NOT FOUND" errors. Save entire text or portion of to disk. Complete file merging capabilities.
MATH - Automatic column totals. Formula computations. User selectable number of decimal positions. Automatically switches to scientific notation when necessary. 14 significant digits.
PRINT CONTROLS - Upper and lower case printing without additional hardware. Automatic paragraph indentation. Single ordouble space printing. Selectable right justification of text. Variable page length and width. Automatic page numbering. Selectable chapter-relative page numbering. Automatic print tabbing. Right or left justification of columnar data. Single sheet or continous form printing. Superscripting and subscripting. Underscoring. Line centering. Automatic link and printing of multiple text files. (48k) 99.95

MICRO INFORMATION SYSTEM™ (48k) $99.95 is a breakthrough in effective information systems of any size. This one system handles accounts payable/receivable, inventories, appointment calendars, cost estimating, real estate listings, sales solicitations, manpower accounting, selective mailings and label printing, dietary information, phone directories and more! On diskette.

U-DRAW II™ (32k) $39.95, a complete graphics package for the Apple II with disk. You can create a figure and rotate, expand, contract or move it anywhere on your video screen with a few simple keystrokes. Save individual figures or complete drawings on disk and recall them later. U-DRAW II automatically builds and edits multi-figure shape tables that are directly transferable to your BASIC programs. You won't find better graphics capabilities at 100 times the price!

APPILOT EDU-DISK™ (32k) $49.95 A complete multi-program C.A.I. system for the APPLE II. Includes program editor and APPILOT interpretor on diskette with extensive on-line HELP lessons plus documentation manual.

THREE MILE ISLAND™ (48k) $39.95 - Is the technology of a nuclear reactor too complex to handle? Now you have the opportunity to decide for yourself, with THREE MILE ISLAND, a realistic simulation of a pressurized nuclear reactor. Four spectacular displays monitor the containment building, turbines, filters, condenser, reactor core and the pump house. Valves, pumps, turbines, filters and control rods are individually activated by keyboard command. The comprehensive documentation describes in detail the operating mechanisms and component interactions.

SUPER-LOAD CASSETTES
U-DRAW (16k) $17.95
ELECTRIC CRAYON (8k) $17.95
MAZE GAME (8k) $12.95
ESCAPE (16k) $12.95
SIDE SHOWS (8k) 12.95
TANK WAR (16k) $12.95
MUSIC BOX (8k) $12.95
BASEBALL (16k)* $14.95
UNCLE SAM'S JIGSAW (32k)* $12.95
GLOBAL WAR (32k)* $17.95
*Plus APPLESOFT Board

MUSE™
THE LEADER IN QUALITY SOFTWARE

Available from dealers or write today to the
MUSE CO., 7112 Darlington Drive, Baltimore, MD 21234
Order by phone (301) 661-8531 MASTERCHARGE and VISA welcome

Nicer Writer

Is screen wraparound a necessary fact of life? Or can the computer adapt to conventional line ending rules? This little BASIC output routine goes a long way toward eliminating wraparound once and for all.

Rick Connolly
41 Roland Drive
Bullwin, MO 63011

Has this ever happened to you: A group of friends are admiring your expensive investment. With the flair of a true computer expert, you press the appropriate buttons, push RETURN, and tell the expectant guests to watch the screen. "Hi!", the computer prints. "Thanks for answering the questions as well as you did, Rick. I can state that you should live 55.215677 more years and have 2.15662 children."

You wait for the applause. Instead, you hear, "How come the words run off the end of the line?" Dead silence. You are embarrassed—for your guests, of course. Instead of seeing the brilliant output of your sophisticated program, your guests saw:

```
HI! THANKS FOR ANSWERING THE QUESTIONS A
S WELL AS YOU DID, RICK. I CAN S ATE THA
T YOU SHOULD LIVE 55.215677 MORE YEARS A
ND HAVE 2.15662 CHILDREN.
```

Now, you and I know that screen wraparound is a fact of life. Perhaps the program concatinated a bit. Or, possibly, it was adapted from ar article written by some thoughtless autl or with a 64 column screen or an 80 column printer. In either case, you probably will soon tire of explaining that nothing is wrong with your magic machine; it just prints funny, sometimes.

This is the wrong approach! We don't adapt to the computer's idiosyncrasies; it adapts to ours. Right? The little subroutine at lines 35000 thru 35010 does a lot to help the wraparound problem. It is a human-oriented subroutine that prints on the screen using much the same rules we would use with a typewriter. Specifically,

It will break a line at a space, comma, period, colon, hyphen, or other character you specify.

If a word is longer than the allowable line, it will be hyphenated (rather arbitrarily, but this is a small subroutine).

At your pleasure, it will indent the first line of the output. This helps increase legibility.

Four variables control the output format. They may be entered once, at the beginning of a program, or they may be changed within the program if required. The variables used are:

CW Column Width. This specifies the maximum columnar width of your output device, and is used for error catching.

M1 Margin indent on the first line printed.

M2 Margin indent on subsequent lines. (Note: Left justification

```
0  REM      NICER WRITER
5  REM
10  REM PROGRAM DEVELOPED
20  REM AND COPYRIGHT (C) 1979
30  REM BY M.R. "RICK" CONNOLLY
     JR
40  REM 5009 RICKWOOD CT NW
45  REM : HUNTSVILLE, AL 35810
46  REM
49  REM
50  REM N$ IS THE STRING TO BE
     PRINTED
60  REM   CW IS THE COLUMN WIDTH
     OF THE PRINTER OR MONITOR
70  REM M1 IS THE TAB INDENTATION
      ON THE FIRST LINE
80  REM M2 IS THE TAB INDENTATION
     ON SUBSEQUENT LINES (TAB 1
     IS 0 INDENTATION ON THE AP-
     PLE)
90  REM M3 IS THE NUMBER OF CHAR-
     ACTERS PER LINE TO BE DIS-
     PLAYED
91  REM
```

```
100 N$ = "THIS IS AN EXAMPLE OF A
      LONG SENTENCE THAT COULD CO
      ME EITHER FROM A PROGRAM WRI
      TTEN FOR A 64 OR 80 COLUMN S
      CREEN OR PRINTER, OR FROM ON
      E THAT CONCATINATES. SUPERC
      ALIFRAGALISTICEXPIALIDOCIOUS
      , NO?"
110 M1 = 5:M2 = 1:M3 = 40:CW = 40

120  HOME : PRINT "PRINTOUT OF ST
     RING N$ AS IT WOULD NOR- MA
     LLY BE PRINTED FROM A PROGRA
     M:": PRINT : PRINT N$: PRINT
     : PRINT
130  PRINT "NICER PRINTOUT OF STR
     ING N$:": PRINT : GOSUB 3500
     0: PRINT : PRINT
140  PRINT "NICER PRINTOUT OF STR
     ING N$ ON LEFT     HALF OF C
     OLUMN:": PRINT :M1 = 3:M2 =
     1:M3 = 20: GOSUB 3500: PRINT
     : PRINT
150  END
```

```
35000  IF M3 - M1 > CW OR M3 - M2
     > CW THEN  PRINT "LINE TO L
     ONG FOR PRINTER.":PRINT :PRINT
     : END : REM  CHECK FOR LINE
     LENGTH
35001 LN =  LEN (N$): FOR I = M3 -
     M1 TO 1 STEP  - 1:BP$ =  MID$
     (NS,I,1): IF BP$ = " " OR BP
     $ = "," OR BP$ = ":" OR BP$ =
     "." OR BP$ = "-" OR LN <  =
     M3 - M1 THEN 35003: REM  FIN
     D BREAK POINT
35002  NEXT I: HTAB M1: PRINT LEFT$
     (N$,M3 - M1 - 1);: PRINT "-"
     :I = M3 - M1 - 1: GOTO 35004
     : REM   HYPHENATE LONG WORD
35003  HTAB M1: PRINT  LEFT$ (N$,
     I): IF LN <  = M3 - M1 THEN
      RETURN
35004 N1$ =  RIGHT$ (N$,LN - I)
35005  IF  LEFT$ (N1$,1) = " " THEN
     LN =  LEN (N1$) - 1:N1$ =  RIGHT$
     (N1$,LN): GOTO 35005: REM
      DELETE EXCESS SPACES
35006 LN =  LEN (N1$):  FOR I = M3
      - M2 TO 1 STEP  - 1:BP$ =  MID$
     (N1$,I,1): IF BP$ = " " OR B
     P$ = "," OR BP$ = ":" OR BP$
      = "." OR BP$ = "-" OR LN <
      = M3 - M2 THEN 35008: REM
     FIND BREAK POINT
```

on the apple is HTAB 1, not HTAB 0).

M3 Length of the line you wanted printed.

**N$** N$ is the dollar string you want nicely printed. You can form N$ through concatination, or can make it equal to another string developed within the program.

The word "Supercalifragalisticexpilidocious" (Does anyone really know how to spell it) is entered in the string N$ of the sample run to point out two characteristics of the nice print subroutine. In the first nice print example, the length of the word has forced it down one line, leaving the preceeding line rather short. In the second example, where the word is longer than the allowable line length, super... is arbitrarily hyphenated. A short line should not appear too often with a 40 column line length, since most words are 10 letters or less in length.

Nicer writer is easy to incorporate into a program, and fast in execution. It will make your programs appear more professional and, best of all, it will keep your friends from asking questions like

```
35007  NEXT I. HTAB M2: PRINT  LEFT$
     (N1$,M3 - M2 - 1);: PRINT "-
     ":I = M3 - M2 - 1:LN =  LEN
     (N1$): GOTO 35009: REM HYPH-
     ENATE LONG WORD
35008  HTAB M2: PRINT  LEFT$ (N1$
     ,I)
35009  IF LN < = M3 - M2 THEN RETURN

35010 N1$ =  RIGHT$ (N1$,LN - I):
      GOTO 35005
PRINTOUT OF STRING N$ AS IT WOULD NOR-
MALLY BE PRINTED FROM A PROGRAM:
THIS IS AN EXAMPLE OF A LONG SENTENCE TH
AT COULD COME EITHER FROM A PROGRAM WRIT
TEN FOR A 64 OR 80 COLUMN SCREEN OR PRIN
TER, OR FROM ONE THAT CONCATINATES.  SUP
ERCALIFRAGALISTICEXPIALIDOCIOUS, NO?
NICER PRINTOUT OF STRING N$:
    THIS IS AN EXAMPLE OF A LONG
SENTENCE THAT COULD COME FROM EITHER A
PROGRAM WRITTEN FOR A 64 OR 80 COLUMN
SCREEN OR PRINTER, OR FROM ONE THAT
CONCATINATES.
SUPERCALIFRAGALISTICEXPIALIDOCIOUS, NO?
```

"Why did it print 'CO MPUTER'?"

MICRO ™ is published monthly by:

MICRO INK, Inc.
34 Chelmsford Street
Chelmsford, Massachusetts
617/256-5515

Second Class postage paid at:

Chelmsford, MA 01824

Postmaster: Send address changes to:

MICRO
P.O. Box 6502
Chelmsford, MA 01824

Publication Number: COTR 395770

Subscription in United States:

$15.00 per year/12 Issues

For subscription and back issue information write to:

MICRO
P.O. Box 6502
Chelmsford, MA 01824
U.S.A.

Subscription Information

Subscriptions are available anywhere in the world, Airmail or Surface. Please write for current subscription rates for your country.

MICRO is carried by distributors in a number of foreign countries. A list of the largest distributors includes:

Computerland Australia Pty, Ltd.
55 Clarence Street
Sydney, N.S.W. 2000
Australia

L. P. Enterprises
313 Kingston Road
Ilford, Essex, England

Micro Shop Bodensee
Markstrasse 3
D-7786 Markdorf
West Germany

The Computer Centre, Ltd.
5345 Woh Hup Complex
Beach Road
Singapore 7

There may be significant differences in price and delivery time between subscribing directly via MICRO INK, Inc. and one of the distributors. Check to determine which supplier is best suited to your individual needs.

Back Issues are generally available for issues number 7 on. The material from issues 1 thru 6 has been reprinted in book form as **"The BEST of MICRO Volume 1"**, and the material from issues 7 thru 12 has been reprinted in book form as **"The BEST of MICRO Volume 2"**. These two books may be ordered directly or may be obtained from your local computer store which carries MICRO.

MICRO™

Disassembling the DOS 3.2

You "Can't tell the players without a score card" and you can not effectively use the Apple II DOS 3.2 without this important information on its organization.

William Reynolds
1733 N. Ford Street
McMinnville, OR 97128

On the surface, DOS 3.2 is identical to DOS 3.1. Upon booting, the DOS is loaded (slave or master), the greetings program is run, MAXFILES defaults to 3, and HIMEM is set at $9600. DOS 3.2 still communicates with the rest of the APPLE via input and output hooks at $36, 37, 38, and 39. (All addresses refer to a 48K machine.)

The differences are many: In Applesoft, DOS does the call 3314 or call 54514 automatically, volume checking is ignored unless explicitly defined in the command, and the system defaults to NOMON C,I,O status. The hooks at $36 and 37 (the print routine) now contain $9E81. The routine to restore DOS is now at $9DBF. This can be called if page 3 is overwritten. The command and error message tables are in different locations. The command table is the same as in the DOS 3.1. The error messages, however, are quite different. After a BLOAD, A$ is now found at $AA72,3; L$ is now found at $AA60,1.

When the keyboard input routine (9E81), is called, DOS checks the mode. If it is in direct mode, the DOS reads the keyboard, then goes to the print routine. The print routine has seven routines of it's own, 0-6. It calls the correct one, depending on whether the mode is direct, deferred, execute, read or write, etc. These routines are all inter-related.

In direct mode, when a return is detected, DOS attempts to match the string in the keyboard input buffer ($200-2FF) to a command in the table. In the print mode, direct or deferred, it stores all characters in the keyboard input buffer until a return is detected. It then checks for a CTRL-D as the first character. If not found, DOS drops out and returns control to wherever it came from. However, if Control D is detected, DOS attempts to match the string to the command table. If a match is not made, it prints "Syntax Error".

When DOS matches a command, it then checks for names, if needed, or numbers, if needed. After getting all data required, a check for optional data is made. After any optional data is read, numbers are changed to hex if need be, the maximum and minimum ranges are compared, then if all data is OK, the number is stored and DOS returns to check for any other optional data.

A routine gets the correct address from the stack, then executes the command. I have highlighted a few of the commands:

PR# and **IN#** do the same function as in BASIC, except that DOS will set the hooks properly before releasing control.

MON and **NOMON** set a mask at $AA74 as follows: 0 = monitor nothing, $10 = monitor 0, $20 = monitor I, $40 = C, and combinations thereof.

MAXFILES resets HIMEM and PP (INT BASIC) and allocates a file buffer via a subroutine at $A7D4.

BRUN does a BLOAD then a JMP ($AA72).

RUN does a load, then jumps to a routine which executes the program.

Which routine is used is dependent upon which language is being used, BASIC, FP RAM, or FP ROM.

LOAD reads the file type and does either INT or FP as needed, then loads the program. When in FP mode, and after the program is loaded, DOS does the call 3314 or call 54514 as needed to set the program pointers for Applesoft.

FP attempts to find a ROM card and turn it on. If possible, it sets the return addresses via a routine at $9D84. If no card is found, the DOS runs Applesoft, then goes to a routine at $9DEA to set return addresses correctly.

INT makes certain the ROM card is off, then goes to $9D84 to set return addresses correctly.

If a person wishes to use DOS from a language or operating system not standard to the APPLE, there is no problem, unless an error is detected. If you do not wish an error message to cause a return to BASIC or Applesoft, the address at $9D5E and F can be changed for your particular system.

Whenever a change in language is done, DOS updates it's return address stack from the stack for that particular language. All commands except PR#, IN#, MON, NOMON, INT, FP (if in ROM), and MAXFILES go through routines that use file buffers.

All commands may be called from monitor or machine language, provided (1) A language change is not needed, (2) the file names have been placed into the name buffer(s), and (3) that any other parameters have been properly placed into their locations as needed.

The disk controller card contains two (2) PROM's, 256 bytes each. One PROM contains the program to start the booting of the DOS. The other is used for a program that, together with some other IC's, actually controls the head position, reading a bit, writing a bit, sending the byte to the APPLE bus, and getting a byte from the APPLE bus. The following locations control the hardware functions. Add 00S0 to each address, S = the slot number of the controller card.

C080-87 These addresses sequentially step the motor that moves the head back and forth. Odd addresses step one way, and even addresses step the other way.

C088 Turns off the drive motor.

C089 Turns on the drive motor.

C08A Enables drive two.

C08B Enables drive one.

C08C,D Control connecting the APPLE bus to the hardware for strobing the byte in or out of the 74LS323 IC shift register, depending upon the previously set status of C08E,F.

C08E,F Read/Write control.

I have documented all routines, subroutines, buffers, and other locations to the best of my ability in the memory maps that follow. Notes tell the function and usage of each. On most items I have given only the starting address. The end address is implied to be the next documented location minus one. On stacks of addresses, the parenthesized number is the number of addresses contained in that stack. Remember that any two-byte items are always stored low byte first. Documentation of addresses in the B000-BFFF area may be in error because that area got too complex for me to retain my sanity.

My thanks to my family for their time and patience, to other persons for their articles on DOS functions, APPLE for their excellent documentation, without which I would have had no idea what was going on, and to Terry and Kent at Computerland of Portland, for use of their printer to obtain 60 feet of hard copy, and their moral support.

APPLE II DOS 3.2 Memory Map

95FF	End of user RAM: HIMEM = 49151
9600	Start of data buffer
9700	Start of track and sector buffer
9800	Start of miscellaneous info buffer
982D	Start of name of file
984B,C	Address of start of miscellaneous info buffer ($9800)
984D,E	Address of start of track and sector buffer ($9700)
984F,0	Address of start of data buffer ($9600)
9851,2	Address of start of name buffer, next file ($0000 = no more files)
9853	Data
9953	Track and sector
9A53	Miscellaneous
9A80	Name
9A9E,F	Address of start of miscellaneous info buffer ($9A53)
9AA0,1	Address of start of track and sector buffer ($9953)
9AA2,3	Address of start of data buffer ($9853)
9AA4,5	Address of start of name buffer of next file down ($982D)
9AA6	Data
9BA6	Track and sector
9CA6	Miscellaneous
9CD3	Name
9CF1,2	Address of start of miscellaneous info buffer ($9CA6)
9CF3,4	Address of start of track and sector buffer ($9BA6)
9CF5,6	Address of start of data buffer ($9AA6)
9CF7,8	Address of start of name buffer of next file down ($9A80)
9CF9 – 9CFF	Unused
9D00	Address of name of first file
9D02	DOS keyin routine address
9D04	DOS print routine address
9D06	Name number 1 buffer address
9D08	Name number 2 buffer address
9D0A	
9D0C	Bottom of DOS
9D0E	
9D10	Address stack for the internal print routines (7)
9D1E	Address stack for the DOS command routines (28)
9D56	Address stack for return to the current language (6)
9D62	Address stack for return to Integer BASIC
9D6C	Address stack for return to Applesoft ROM (6)
9D78	Address stack for return to Applesoft Disk (6)
9D84	(3D3G) Control B, re-enters INT or FP (ROM only)
9DBF	(3D0G) Restores DOS and re-enters current language
9DEA	Restores $3D0 – $3FF from $9E51 – $9E80
9E51	Stack for the above routine
9E81	Keyboard input routine
9EBD	Calls correct internal print routine, depending upon mode
9ED1	Restores keyboard and print hooks
9EEB	Internal routine for information from the disk
9F12	Internal routine for printing
9F23	Prints and exits DOS
9F2F	Keyboard input internal routine
9F52	Internal routine for sending information to disk
9F61	Routine to correct internal routine
9F71	Used by the EXEC command
9F83	Mask MON status, print and exit

9FC8	Does a RETURN
9FCD	Start of section that attempts to match to a command and get all information needed and all optional information given. Checks syntax and ranges before execution.
A229	PR# routine
A22E	IN# routine
A233	MON routine
A23D	NOMON routine
A251	MAXFILES routine
A263	Start of DELETE routine
A271	Start of LOCK routine
A275	Start of UNLOCK routine
A27D	Start of VERIFY routine
A281	Start of RENAME routine
A298	Start of APPEND routine
A2A3	Start of OPEN routine
A2EA	Start of CLOSE routine
A331	BSAVE routine
A35D	BLOAD routine
A38E	BRUN routine
A397	SAVE routine
A413	LOAD routine
A4D1	Run routine
A4E5	Runs Integer BASIC program
A4F0	CHAIN routine
A4FC	Runs FP ROM program
A506	Runs FP RAM program
A510	WRITE routine (set up)
A51B	Read routine (set up)
A54F	INIT routine
A56E	Catalog routine
A57A	FP routine
A59E	INT routine
A5C6	EXEC routine
A5DD	Position routine
A60E	Starts the read process
A626	Starts the write process
A644	Stores data coming from text file into keyboard buffer. Used by the EXEC command.
A65E	Error checking?
A679	Closes files, exits DOS
A682	Goes to hardware routines
A69D	Sets up address of name section of next file
A6AB	Close the buffer last used
A6C4	Prints, "SYNTAX ERROR "
A6C8	Prints, "NO BUFFERS AVAILABLE"
A6CC	Prints, "PROGRAM TOO LARGE"
A6D0	Prints, "FILE TYPE MISMATCH"
A6D5	Prints other error messages by message number contained in $AA5C
A71A	Moves parameters given to locations for use by hardware routines
A743	Moves name from the name buffer to the name section of the file buffer
A74E	Moves addresses of sections of file buffers to locations for use by hardware routines
A764	Attempts to find a file buffer already in use by the name given
A74F	
A7C4	Checks file type
A7D4	Sets up file buffers and addresses (used by MAXFILES)
A851	Restores DOS hooks
A884	Start of command table
A909	This is a table of two-byte masks. One byte is used to determine what type of extra data is needed by a command. The other byte is used by the hardware routines for what file type to create or look for.
A941	Table containing the letters V, D, S, L, R, B, A, C, I, O. This is used when checking for optional data.
A94A	Table of bytes for determining what type of optional data to look for.
A995	Table of minimum and maximum ranges for V, D, S, L, R, B, A.
A971	Start of error message table
AA3F	Relative address of start of error message, i.e. ($A971,X)
AA4F,50	Address of name section of next available file buffer
AA51	
AA52	Internal print routine number
AA53,4	PR# hooks out of DOS
AA55,6	IN# hooks out of DOS
AA57	Number of total file buffers
AA58	Number of file buffers not in use
AA59 –	Temporary storage used by various routines
AA5E	Mask for MON and NOMON
AA5F	Command number
AA60 – 61	Found L$ from a BLOAD
AA62 – 65	Temporary storage used by various routines
AA66,7	Defined volume number
AA68,9	Defined drive number
AA6A,B	Defined slot number
AA6C,D	Defined length
AA6E,F	Defined record number
AA70,1	Defined byte number
AA72,3	Defined address
AA74	
AA75	Start of file name buffer number 1
AA93	Start of file name buffer number 2
AAB1	
AAB2	Control D
AAB3	Mode (direct, deferred, etc.)
AAB4,5	
AAB6	Value used for language, e.g. INT = 0, FP RAM = C0, FP ROM = 80
AAB7	
AAB8	The name, "Applesoft"
AAC1,2	Address of start of IOB (used by RWTS)
AAC3,4	Address of start of buffer for track/sector list (used by RWTS)
AAC5,6	Address of start of buffer for data (used by RWTS)
AAC7,8	Top of total RAM in the APPLE II
AAC9	Address stack for hardware routines (14)
AAD5	Address stack for hardware routines (6)

AAF1 Address stack for hardware routines (6)

AAFD Goes to the correct hardware routine

AB28 Reads VTOC and reads directory attempting to find an entry with the same name as the one given. If not found, checks the table of masks to see if it is allowed to create a file. If it may, it does so, and if not, it exits with "FILE NOT FOUND" or "LANGUAGE NOT AVAILABLE"

ABDC Clears miscellaneous info hardware buffer; sets volume number, drive number and slot number.

AC06 Close routine. Updates VTOC, track bit map, and sector count of directory entry as needed.

AC3A Rename routine. Finds directory entry, stores new name in entry, then writes that directory sector back to disk.

AC58 Goes to correct hardware routine

AC70 Goes to correct hardware routine

AC87 Sets parameters for following routine

AC8A Actually reads text file

AC93 Sets parameters for following routine

AC96 Reads program or binary file

ACA8 Puts byte being read into buffer

ACBB Sets parameters for following routine

ACBE Writes into text file

ACC7 Sets parameters for following routine

ACCA Writes program or binary file

ACDA Gets byte being written from buffer

ACEF Lock hardware routine

ACF6 Unlock hardware routine

AD12 Sets parameters for following routine

AD18 Verify hardware routine

AD2B Delete hardware routine

AD54 Part of delete routine, frees sectors used by deleted file.

AD98 Catalog hardware routine

AE42 Part of catalog, prints the number in $44 as three digit ASCII.

AE6A Moves miscellaneous info from the file buffer to the hardware buffer.

AE7E Moves miscellaneous info from the file buffer to the hardware buffer.

AE8E Initialize hardware routine

AF08 Sets 42 and 43 as pointers to sections of the file buffer

AF1D Writes data section of file buffer to disk

AF34 Writes track/sector list section of file buffer to disk

AF4B Sets hardware pointer to the track and sector list section of the file buffer being used

AF5E Checks position in file. If out of current sector, reads/writes next sector, updates VTOC buffer, updates track/sector list section of file buffer if in write mode.

AFDC Reads from disk into data section of file buffer

AFE4 Sets hardware pointers to data section of file buffer being used

AFF7 Reads VTOC to its buffer ($B3BB – B4BA)

AFFB Writes VTOC from its buffer

B011 Reads a directory sector into its buffer ($B4BB – B5BA). Initially reads sector A, successive entries into this subroutine read successive sectors from the disk. When all sectors have been read and the subroutine is called again, it will merely exit with the carry set.

B037 Writes current directory sector from buffer to disk.

B052 Sets up IOB for directory sectors, goes to RWTS

B0A0 End of above if no error

B0A1 Start of error handling routine for above

B0B6 Checks position in file, reads/writes next sector as needed

B134 Initializes data section of file buffer to all zeroes

B15B Sets next position in file

B194 Increments position in file

B1A2 Sets next RAM address

B1B5 Calculates how much RAM is left

B1C9 Reads VTOC and successive entries, attempting to find the specified file name.

B21E Puts name of file into directory

B224 Sets next sector, updates VTOC buffer

B2C3 Updates VTOC

B2DD Calculates track bit map for VTOC

B300 Sets/checks parameters for file?

B35F Routine with different entry points to exit the hardware routines with error

B397 – A6 Temporary storage for hardware routines

B3A7 – AA T, I, A, B Used by catalog for file types

B3AB,C

B3AD – BA In reverse order, the string, "DISK VOLUME"

B3BB VTOC buffer

B4BB Directory buffer

B5BB – D0 Temporary storage for hardware routines

B5D1 – FF Miscellaneous info section of currently used file

B600 Buffer. Purpose?

B700 Reads drive 1, current slot, $B1 sectors, track 0, sector A into RAM starting at $1B00. Boot routine?

B74A Writes $0A sectors, starting from $B600, then $1B sectors, starting at $1B00, beginning at track 0 sector 0.

B793 Increments track/sector as needed and data address for above two routines

B7B5 Calls RWTS, checks status upon return

B7C2 Sets address of data buffer, and sets expected volume number

B7DB Stores zeroes in one page, starting at the address in $42, 43

B7E7 Start of IOB and device characteristics table

B800 Part of RWTS?

BA90 – FF Temporary storage for RWTS?

BB00 One-page buffer (RWTS?)

BC00 One-page buffer (RWTS?)

BD00 Start of RWTS

BFD4 End of RWTS

BFD5 Various endings sections for the hardware routines

BFFF End of RAM

Hooking PET to Ma Bell

The dream of many microcomputerists to use their system as a terminal connected to a large computer system can become a pratical reality. The $50.00 hardware for any 6502 based system, and the software for a PET, are fully described.

C.H. Scanlon
P.O. Box 22
Arkansas State University
State University, AR 72467

Having worked with my 8K PET for almost a year, I have become hooked on microcomputers and am enjoying learning and experimenting with a great machine. Like most microcomputer enthusiasts, I dream of more memory, disks, printers, etc. However, attempting to raise a family on a teacher's pay means that I have limited funds. So I wired up a PET to RS-232 modern interface, plugged into a modem, and bingo — by dialing up the computer system on the campus of Arkansas State University, I have all of these plus much more hooked to my PET. If you have telephone access to a computer system or a friend with an answer modem on his computer, here is the hardware and software to get you started communicating on the telephone.

The interface shown in Figure 1 can be built for under $50 including connectors, wiring, etc., and can be plugged into any RS-232 modem (I have a U.S. Robotics Model 310 which lists for $149). A TTL compatible modem can be wired directly to pins 2 and 6 of the MC6850. All the parts, except the crystal, are fairly common and can be ordered from most mail order electronics parts firms. The 1.229 megaHertz crystal can be ordered from any crystal manufacturer for around $10. This interface can be connected to any 6502 or 6800 based microcomputer that allows direct access to the microprocessor bus, for example, the APPLE, KIM, SYM, SWT, OSI, etc.

The software is written in BASIC and makes the PET act like a TTY type "dumb" terminal. The control characters are obtained by using the shift key. For example, control S is simply shift S. Although this program appears to limit the PET, it really doesn't since you can hit the stop key, write and run a program in the extra RAM and get back to the terminal program with a RUN 190 or a GOTO 190. For example, you could write a BASIC program starting at line number 500, compute a bunch of data, POKE the data to the modem, and then return to the terminal program with a GOTO 190.

```
 10 REM              TERMINAL PROGRAM
 20 REM              BY C.H. SCANLON
 30 REM              P.O. BOX 22
 40 REM              STATE UNIVERSITY, ARKANSAS
 50 REM              72467
 60 REM
101 DATA   173, 246, 191, 48, 3, 76, 133, 230, 173, 247,
           191, 88, 41, 127, 170, 169, 32, 172, 226, 0
102 DATA   145, 224, 138, 32, 210, 255, 169, 160, 172,
           226, 0, 145, 224, 76, 133, 230
105 FOR I = 826 TO 861: READ X: POKE I, X: NEXT
110 POKE 537, 58: POKE 538, 3
115 POKE 49142, 3
120 POKE 59468, 14
130 PRINT "(cs)  * * * * TERMINAL * * * *"
140 PRINT "(cd)(cd) Type RUN 190 but don't hit the return yet".
150 PRINT "(cd) Dial 935-9372 and wait for the tone".
160 PRINT "(cd) Place receiver in holder and hit return".
180 STOP
190 POKE 49142, 129
195 FOR I = 1 TO 30: NEXT: POKE 49143, 7
200 GET A$: IF A$ = "" GOTO 200
210 IF ASC(A$) = "shift S" THEN PRINT "(cs)"
215 IF ASC(A$) < 192 GOTO 300
220 IF ASC(A$) > 244 GOTO 300
225 POKE 49143, ASC(A$) - 192: GOTO 200
300 POKE 49143, ASC(A$): GOTO 200
```

NOTE: (cs) means clear screen and (cd) means cursor down.

Figure 2

Software

The MC6850 Asynchronous Communications Interface Adapter (or, in the buzz words of computerland, simply speak the letters A-C-I-A) is located in page B and has multiple addresses. I use hex BFF6 = 49142 as the address to POKE to the control register and to PEEK at the status register. Address BFF7 = 49143 is used to POKE a byte to the modem and to PEEK at a byte from the modem.

The BASIC program provides directions for the operator, data transfer from the modem to PET, data transfer from PET to the modem, and miscellaneous programming needs.

Lines 101 – 105 POKE a machine language routine into the second cassette buffer, and line 110 POKEs the

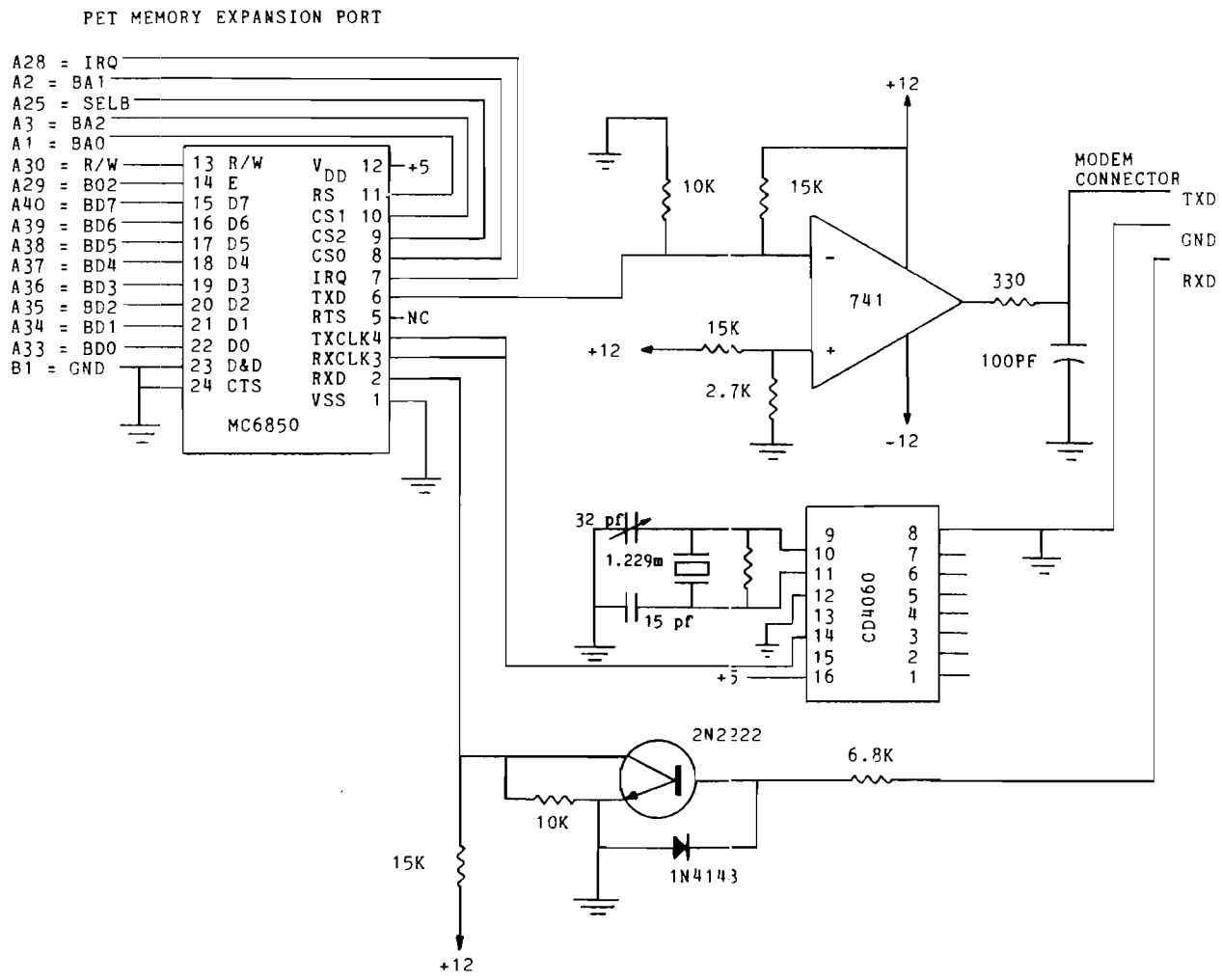


Figure 1

address of this routine into the interrupt address location of RAM so that, when an interrupt occurs, PET will check the ACIA before it checks the other possible interrupt locations.

When the ACIA receives a serial byte from the modem, it strips off the start and stop bits, converts it to parallel, and then interrupts the CPU. PET is then routed to the routine beginning at 033A.

Lines 033A – 033C transfer the contents of the ACIA status register, to register X and lines 033D – 033E cause a branch if bit 7 is set, indicating that the ACIA did interrupt the CPU and has a byte to transfer. Otherwise, lines 033F – 0341 transfer PET to the ROM interrupt sequence. Lines 0342 – 0344 transfer a data byte from the ACIA to register A and line 0345 clears the CPU interrupt to allow for other interrupts. Lines 0346 – 0347 strip the parity bit from the data byte and line 0348 transfers it to register X for temporary storage.

Next, to erase the cursor, lines 0349 – 034A load A with $20 (note that CHR$(32)is a blank). Lines 034B – 034D get the current position of the cursor on the video line and lines 034E – 034F then clear the cursor.

To type the character, line 0350 retrieves the data byte from register X and stores it in register A and lines 0351 – 0353, then types the character in the next print position.

To set the cursor, lines 0354 – 0355 load register A with $A0 (note that CHR$(160) = reverse blank), lines 0356 – 0358 get the current position of the cursor on the video line, and lines 0359 – 035A then set the cursor.

Lines 035B – 035D then transfer control back to the PET interrupt routine. Back in the BASIC program, line 115 POKEs 3 into the ACIA control register which then resets the ACIA. Line 120 sets the lower case letter mode and then lines 120 – 180 print instructions and stop.

Since the answer modem at Arkansas State University uses seven bits plus parity plus two stop bits, line 190 programs the ACIA to transfer data in this mode. Reference 1 explains how to program other modes. Also, since the Arkansas State University computer initially waits for a control G, line 195 has a delay and then POKEs a 7 = ASCII CTRL-G to the modem. Lines 200 – 300 then wait to get a character from the keyboard, convert the character to ASCII, and POKE it to the ACIA.

Hardware

The MC6850 is wired directly to the CPU bus through the memory expansion port. I use page B by wiring CS2 to SELB. Details of programming the ACIA can be found in reference 1.

The 1.229 megaHertz cystal and the C4060 counter put out a 4800 Hertz square wave to the ACIA. The ACIA further divides it by 16 to obtain a baud rate of 300. Reference 2 indicates how to get

other baud rates. The 741 op amp converts the RS-232 logic from the modem to TTL as described in reference 3.

You will need a ±12 and +5 volt power supply. If you use a TTL compatible modem, you won't need the ²12 volt supply and you can get +5 volts from the second cassette port.

Questions

There are lots of software questions I have not answered. For example, how can a program be copied directly from the cassette to the modem? How can a program or data file be "saved" by sending it to the storage facilities at the other end of the line and how can it be retrieved later? With the exception of displaying more characters, what can an expensive "smart" terminal do that PET can't do? As I stated earlier, this article is merely a start.

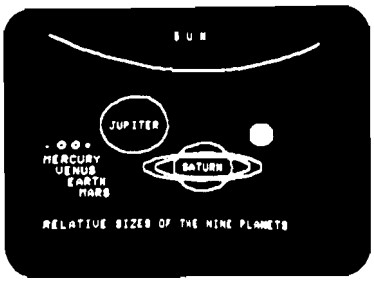
```
MICRO-WARE ASSEMBLER 65XX-1.0 PAGE 01

0010:
0020:
0030: 033A                      ORG   $033A
0040: 033A AE F6 BF             LDX   $BFF6  GET STATUS REGISTER OF ACIA
0050: 033D 30 03                BMI   $0342  BRANCH IF BIT 7 SET
0060: 033F 4C 85 E6             JMP   $E685  JUMP TO PET INTERRUPT
0070: 0342 AD F7 BF             LDA   $BFF7  GET BYTE FROM ACIA
0080: 0345 58                   CLI          CLEAR INTERRUPT FLAG
0090: 0346 29 7F                ANDIM $7F    STRIP OFF PARITY BIT
0100: 0348 AA                   TAX          STORE THE BYTE
0110: 0349 A9 20                LDAIM $20    CHR(32) = BLANK
0120: 034B A4 E2                LDY   $00E2  GET CURSOR POSITION ON LINE
0130: 034D 91 E0                STAIY $E0    CLEAR CURSOR
0140: 034F 8A                   TXA          RETRIEVE THE BYTE FROM X
0150: 0350 20 D2 FF             JSR   $FFD2  TYPE IT AS A CHARACTER
0160: 0353 A9 A0                LDAIM $A0    CHR(160) = REVERSE BLANK
0170: 0355 A4 E2                LDY   $00E2  GET CURSOR POSITION ON LINE
0180: 0357 91 E0                STAIY $E0    SET CURSOR
0190: 0359 4C 85 E6             JMP   $E685  JUMP TO PET INTERRUPT
ID=
```

Figure 3

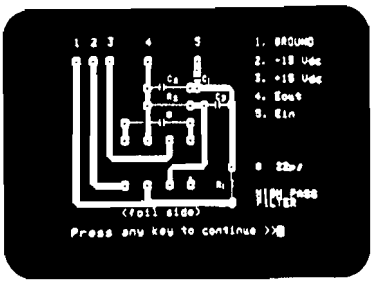
References

1. *An Introduction to Microcomputers Volume II*, by Osborn, Jacobson, and Kane, Adam Osborne and Associates, Incorporated.
2. *CMOS Cookbook* by Don Lancaster, Howard W. Sams and Company, Incorporated.
3. *Peripheral Interfacing* by Rod Hallen, *KILOBAUD Microcomputing*, June 1979.

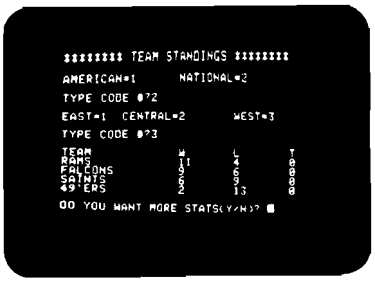
Software for the APPLE II



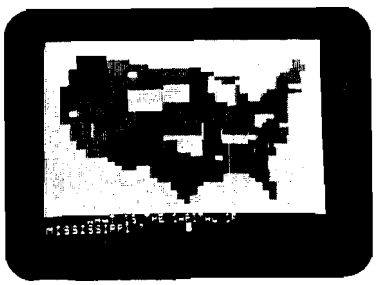
THE PLANETS $15.95



ACTIVE FILTERS $24.95



FOOTBALL PREDICTIONS $19.95



STATE CAPITALS $9.95

FORMAT

PROGRAMMA's FORMAT (Version 1.Ø) is a command oriented text processor designed to be fully compatible with PIE (PROGRAMMA Improved Editor).

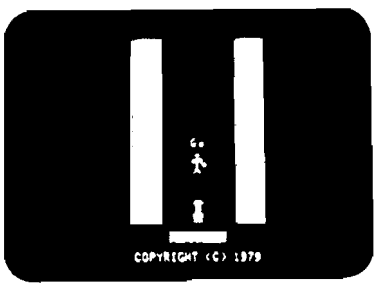
FORMAT's system of imbedded commands (within the text) give it an ease of operation similar to text formatters found on some mini-computers.

FORMAT features right margin justification, centering, page numbering, and auto-paragraph indent.

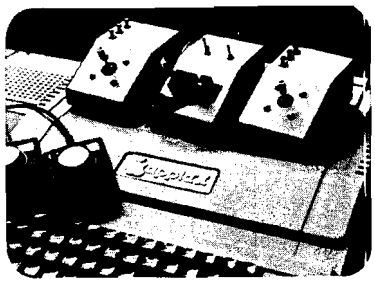
The following commands are available with FORMAT:

.ad		Begin adjusting right margins
.bp	n	Begin page numbered n
.br		Cause a line break
.ce	n	Center next n lines without fil
.fi		Start filling output lines
.fo	t	Foot title becomes t
.he	t	Head title becomes t
.in	n	Indent n spaces from left margin
.li	n	Literal, next n lines are text
.ll	n	Line length including indent is n
.ls	n	Set line spacing to n
.ml	n	Top spacing including head title
.m2	n	Spacing after heading title
.m3	n	Spacing before foot title
.m4	n	Bottom spacing including foot title
.na		Stop adjusting right margins
.nf		Stop filling output lines
.pl	n	Page length is n lines
.pp	n	Begin paragraph= .sp, .fi, .ti n
.sp	n	Space down n lines, except at top
.ti	n	Temporary indent of n
.ul	n	Underline next n input lines

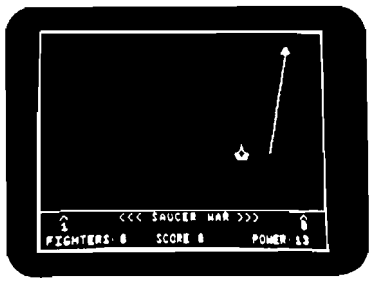
All orders include 3% postage and handling.
Apple II is a registered trademark of Apple Computer, Inc.
California residents add 6% Sales Tax
VISA & MASTERCHARGE accepted.



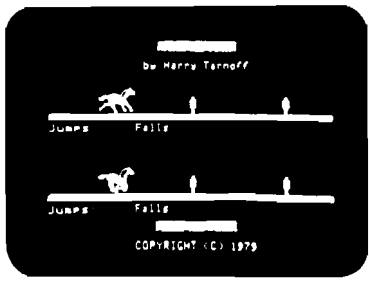
SPEEDWAY $15.95



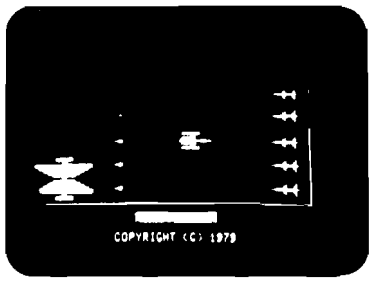
JOY STICK $49.95
EXPAND-A PORT $49.95



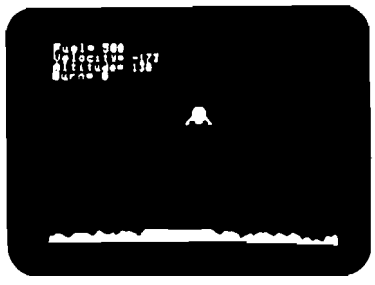
SAUCER WAR $15.95



CANTER DOWNS $15.95



BATTLESTAR I $15.95



LUNAR LANDER $9.95

PROGRAMMA INTERNATIONAL, Inc.
3400 Wilshire Blvd.
Los Angeles, CA 90010
(213) 384-0579
384-1116
384-1117

Dealer Inquiries Invited

PROGRAMMA Software Products

Spelunker

Spelunker is not for the faint of heart! It presents many interesting and useful programming techniques in the pleasant format of a game. As you play, keep telling yourself "It's only a game, it's only a game ..."

Thomas R. Mimlitch
1547 Cunard Road
Columbus, OH 43227

This is an adventure fantasy series in which you become directly involved in exploration of a mysterious cavern in southwest Kentucky called Devils' Delve. If you have never played before, you should take a guide along. The guide will read the chamber descriptions as you enter each room for the first time. He can also supply some hints and clues to help you when you are stuck. Only the guide should use the room descriptions, word lists, and the map of the caverns. However, younger players may need some of these aids to help them.

Spelunker is an interactive game. You must converse with the program in order to explore the caverns and locate their treasures. You can talk in sentences, if you wish; but the program will use only one verb and one noun to establish meaning. For this reason, it is best to converse in verb/noun phrases. In the case of moving from chamber to chamber, for example, enter "GO W" or simply "W" and the verb "GO" will be implied. The Spelunker program will move you into the next room to the west upon receiving this command. Other examples might include "TAKE LIGHT" or "JUMP DOWN".

With this brief introduction you should be ready to explore the caverns of Spelunker. While you are about it, try drawing a map of the cave. You may also wish to discover exactly what vocabulary is understood by the program. The material that follows is for the guide only — so don't ruin your first adventure by peeking at it.

For the Guide Only

In the 16K APPLE II version of Spelunker, the chamber descriptions are not part of the program because of limited memory size. These room descriptions have been prepared for the adventurer's guide. The guide may read each room description as the adventurer enters the chamber for the first time.

1. **Mouth**: You are at the mouth of a large cavern. The sides of the entrance slope steeply upward, and a mysterious passage leads west into the cave.
2. **Tree room**: A towering, withered tree stands in what appears to be a dried up river bed. From it you seem to hear echoing sounds saying, "Water...water...water..."
3. **Writing room**: Do not read this description if the room is dark. The writing room is a large, oval chamber with tall ceilings and massive stalagmites. The smooth eastern wall has some writing on it — cryptic characters that spell out, "THE SPIRITS OF THE FRUIT."
4. **Pit room**: A small chamber with an immense stalagtite hanging from the center of the ceiling, directly over the mouth of a bottomless pit.
5. **South lake shore**: You stand at the edge of a misty lake that stretches endlessly out before you to the north.
6. **West lake shore**: You are standing on a damp, sandy shoreline with a very low passage leading off to the west. A clammy draft issues from the low-ceilinged passage.
7. **North lake shore**: A small, sandy beach on the northern edge of Misty Lake.
8. **Maze room**: Also known as the swiss cheese room. You loose your sense of direction because twisting passages are coming and going at all points of the compass.
9. **Frozen river room**: What appears to be a petrified river bed slopes gently upward leading toward he west. It has a low, four-foot ceiling.
10. **Swift river room**: You hear swiftly running water, as you enter this room, and you see a narrow, churning, underground river flowing to the south.
11. **Hub room**: A magnificently decorated chamber with crystaline designs and intricate rock formations. A narrow, fast moving river flows through the hub room.
12. **Ice room**: Mysteriously, ice forms very quickly in this chamber, encapsulating anything left there for too long. There is so much ice that you can't even get into the room; however, you see an exit on the other side of the chamber.
13. **Chimney room**: A small, smoke filled chamber with a fire burning in a natural fireplace in the north wall. Apparently, a chimney leads far up through the rock and out of the cavern.
14. **Gold room**: As you enter this room, the first thing that you notice is a pile of golden treasures nestled into a nook on the far side. Before you take another step, a foul-smelling ogre jumps out from a hole in the side wall and rushes forward to protect his gold.
15. **Bones room**: Lining the walls of this chamber are the skeletons of pirates long since dead. An omnious curse is uttered by all of the skeletons in unison, as you enter the room, and the curse shadows your travels throughout the cavern.
16. **Bat room**: The ceiling is all but invisible for the tens of thousands of bats sleeping there. In one corner of this room lies an old, rusted chest. As you open the chest, the bats begin to stir. Inside the chest is a king's ransom in jewels: diamonds, rubies and emeralds.
17. **Ghost room**: An eerie feeling of demonic power lurks in this chamber.
18. **Misty Lake**: You are in the middle of Misty Lake. A strange glow emanates from the bottom of the lake. You turn off your light and notice an enormous, bright pearl nestling inside a gigantic clam. The clam is at the bottom of the lake, in only ten feet of water.
19. **Swift River**: This narrow, fast flowing river is outside the cavern. It runs south for a few yards and then disappears underground.

Having been exposed to a fantasy program called *Adventure* which seems to reside on many large timesharing networks, I was challenged to see if this type of game could be handled on a micro. Thus the dream stage began. I thought up monsters, treasures, a cave structure, tools, tricks and battles. The major goals emerged:

Pseudo-English input commands (verb-noun phrases)

Interconnected rooms one could travel through

Objects one could take, put, carry and use

Monsters / treasures; do battle, take rewards

Secrets to be discovered

The obvious method was to tabularize as much data as possible so that similar functions could be implemented as subroutines. This left only special handling routines to be added.

The program was organized into five major sections. Lines numbered 30xxx initialize the tables and variables. Lines numbered 4xxx to 10xxx print out the current location and status for the player. Lines numbered 1xxx read and decode the input string. Lines in the 2xxx range perform the command action, if possible. In lines with 3xxx numbers the monsters have an opportunity to react to their environment. Each of these sections was developed, tested and integrated separately from the others.

Table 1: Sample word tables for the guide.

The following lists of verbs and nouns are for use if you are having difficulty in communicating with Spelunker. Not all of these words have meaning or utility in this adventure. I didn't want to make it *too* easy!

VERBS

BITE	CARRY	CLIMB	DIG	DRINK	DRIVE
DROP	EAT	FIGHT	GO	HELP	HIT
JUMP	KILL	PUT	RUB	RUN	START
STOP	TAKE	THROW	USE	WALK	WISH

NOUNS

APPLE	AX	BATS	BOMB	BONES	CAVE
CHEST	CLAM	CURSE	DOWN	E	FIRE
GHOST	GOLD	ICE	KNIFE	LAKE	LAMP
LIGHT	N	NE	NW	OGRE	PEARL
RAFT	RIVER	ROPE	S	SE	SW
TENT	TREE	TRUCK	UP	W	WATER

Input commands

A list of verbs and nouns was developed and categorized as to nature or function. After entering these tables into the program, I worked on the routine to read and decode input commands. Each word was picked out of the input string, then searched for in the noun and verb lists. The first recognized verb and noun numbers were the output of this routine, and this output controlled the action routines. I later added an edit to compare the noun type and verb type to see if they were compatible.

Objects to take and put

Parallel to the noun list is the status list which gives the room number where an object currently resides. A -1 indicates that the object is in the possession of the player. In the output section, objects in the current room (LOC) were printed and the objects in the players possession were also reported. The second action routine was added next — the TAKE and PUT routine. TAKE changed the status of a noun to −1, while PUT set its status equal to LOC. Again I tested the program and played with it, moving things all over the caves.

Verb Table

	Verb	Type	Sensative Noun Types						
			Direction	Location	Weapon	Monster	Treasure	Tools	Foods
1	GO	1	x						
2	JUMP	11	x	x		x			
3	RUN	1	x						
4	WALK	1	x						
5	DRIVE	1	x						
6	CLIMB	3	x	x					
7	DIG	2		x					
8	CARRY	116			x		x	x	x
9	DROP	116			x		x	x	x
10	PUT	116			x		x	x	x
11	TAKE	116			x		x	x	x
12	USE	36			x			x	
13	WISH	36			x			x	
14	THROW	4			x				
15	HELP	8							
16	KILL	8				x			
17	STOP	40					x		x
18	HIT	8					x		
19	FIGHT	8				x			
20	RUB	16					x		
21	START	32					x		
22	DRINK	64							x
23	EAT	64							x
24	BITE	64							x

Cave room structure

The map was finalized, giving each room a number. The interconnections were entered into the N, E, S and W arrays, with a positive number indicating an exit in that direction to the room number specified. A series of statements were inserted in order to print out the current room descriptions, but at the time only the room name was printed. Later I discovered that there was not enough memory to put in the complete descriptions in any event.

The first of the action routines — the MOVE routine — was coded next. If there was a possible move in the requested direction, the LOC variable was set to the new room and its description was printed. This portion was a lot of fun to test and debug.

Monsters, treasures and battles

The monsters and treasures were merely noun objects in the caves, like all of the other things. A relationship was defined between the monster, his treasure, the player, and the player's use of weapons. Thus grew up the monster table and the weapons table. The monster table identifies the monster, determines his strength, defines his treasure, identifies his home chamber, and determines how quickly he moves about the caves. The monsters move through the caverns to find their treasures if they are stolen. In the table are certain base probability factors for the monster to kill the player, steal all the player's treasures, or steal only the treasure than originally belonged to the monster.

The weapons table details the power of each of the player's weapons and determines which monsters they are effective against. The next action routine was ready to implement: the ATTACK routine. This is invoked whenever a weapon is used, put, thrown, and so on. Any monsters in the room are attacked, and their life forces are decreased by a random amount limited by the force of the weapon used. When a monster's life force is reduced to zero, it is eliminated.

Of course, it is not fair to let the player cut the demons to shreds without allowing them to fight back. Thus came the REACTION routines. Happy monsters are those that have their own treasures in their room and have not been attacked. Any monsters that are not happy will seek someone to vent their anger upon, and that person is the player. A very intricate set of probabilities decides the outcome of this anger. The more the monster has been hurt by the player's attacks, the weaker his counterattack will become. But also, the more times he has countered in vain, the madder he gets! Nothing is more deadly than a mad monster.

Noun Table

Noun		Type	Status (Location)
1	N	Direction	0
2	NE	Direction	0
3	E	Direction	0
4	SE	Direction	0
5	S	Direction	0
6	SW	Direction	0
7	W	Direction	0
8	NW	Direction	0
9	UP	Direction	0
10	DOWN	Direction	0
11	CAVE	Location	0
12	LAKE	Location	0
13	RIVER	Location	0
14	TREE	Location	0
15	AX	Weapon	4 = Pit
16	BOMB	Weapon	3 = Writing
17	CURSE	Weapon	15 = Bones
18	FIRE	Weapon	13 = Chimney
19	KNIFE	Weapon	1 = Mouth
20	CLAM	Monster	18 = Misty Lake
21	BATS	Monster	16 = Bat
22	BONES	Monster	15 = Bone
23	GHOST	Monster	17 = Ghost
24	OGRE	Monster	14 = Gold
25	CHEST	Treasure	16 = Bat
26	GOLD	Treasure	14 = Gold
27	PEARL	Treasure	18 = Misty Lake
28	LAMP	Treasure	12 = Ice
29	RAFT	Tool	5 = South Shore
30	ROPE	Tool	9 = Frozen River
31	TENT	Tool	1 = Mouth
32	TRUCK	Tool	1 = Mouth
33	LIGHT	Tool	1 = Mouth
34	WATER	Food	0
35	APPLE	Food	0
36	ICE	Water	12 = Ice

Room Table

Room		Tunnel Connects				Notes
		N	E	S	W	
1	Mouth	50		19	2	Truck Tent Knife Light
2	Tree		1	3		
3	Writing	2		10	20	Bomb
4	Pit		20			Ax Use rope to go down
5	South Lake Shore	-18				Raft-north Rope-up
6	West Lake Shore		-18		12	Raft-east
7	North Lake Shore	9		-18		Raft-south
8	Maze	8	9	8	20	All 45's return to Maze
9	Frozen River	7	1		8	Rope
10	Swift River Room	3		-11		Raft-south
11	Hub	13	14	-49	21	- 15 22 12 (NE SE SW NW)
12	Ice			11	6	Ice Lamp
13	Chimney			11		Fire Rope-up
14	Gold				11	Gold Ogre
15	Bones				11	Curse Bones
16	Bats	22				Chest Bats
17	Ghost		21			Ghost
18	Moosty Lake	7		5	6	Pearl Clam
19	Swift River	1				
20	Intersect 1	8	3		4	
21	Intersect 2		11	22	17	
22	Intersect 3	11		16	21	
49	Falls (Over)					Death
50	Home					End game

Lots of testing and refinements later, SPELUNKER took its maiden voyage. Surely a program like this is never finished. The framework has been laid for all sorts of adventures, whatever one can imagine. And, now that I have more memory, I can expand the scope and capabilities of the program.

Monster Table

Monster name	Ogre	Bats	Ghost	Clam	Ice	Bones
Monster number	24	21	23	20	36	22
Reward	Gold	Chest		Pearl	Lamp	
Reward number	26	25	0	27	28	0
Move delay	0	0	0	1	1	1
Move increment	2	4	6	0	0	0
Attack count	0	0	0	0	0	0
Kill probability	60	60	0	90	0	0
Steal all probability	30	40	0	60	60	0
Steal own probability	55	90	0	65	0	0
Home room number	14	16	17	18	12	15
Life force quotient	100	40	50	60	25	75

Weapon Table

Weapon name	Ax	Bomb	Fire	Knife	Light	Ice
Weapon Number	15	16	18	19	33	36
Power	100	150	30	50	30	40
Attacks Monster No. 1	24	24	21	24	23	21
Attacks Monster No. 2		22	22	20		
Attacks Monster No. 3		36	36			

Program Flow

Initialize - 30000's
Dimension and set up data for nouns, verbs, noun types, verb types, status or location of nouns, noeth, south, east and west tunnel connections, monster life force tables, and weapons table.

Output - 4000's
Print room descriptions, possible exit directions, and room contents as well as your possessions.

Input/Decode - 1000's
Read your typed-in command, select each word and scan it against the noun and verb lists.The first valid noun and verb are edited and used to control the rest of the program.

Actions - 2000's
This routine handles takes and puts, special verbs and nouns, your attacks on life forces, and movement through the caverns.

Reactions - 3000's
The demonic life forces who have been attacked or who do not have their own treasures fight back. Based on complex probabilities, they either kill, steal your treasures, or wander the caverns in search of you.

Microbes

Move It; Relocating PET Source Programs and Object Code
Herman, 16:17

The following table should have appeared with Harvey B. Herman's article in MICRO 16:17 MOVE IT ...

Operation	Comment
1. Load "MONITOR" and rewind	Sets up PET PARAMETERS
2. SYS 62894	Load tape header
3. SYS 1039	Run Monitor
4. M 027B, 027B	Display tape address
027B 00 04 6B 07	
5. 027B 00 1C 6B 1F	Change addresses
6. X	Return to BASIC
7. SYS 62403	Finish monitor move
8. POKE 135,28	Protect relocated monitor
9. LOAD "MODIFY" and RUN	Run BASIC MODIFY program
10. LOAD "MONITOR" and RUN	Run MONITOR program to finish relocation

Figure 1: Monitor Relocation Procedure

AIM-65 In the Ham Shack
De Jong, 16:29

The following table should have appeared with Marvin L. De Jong's article. It is a table of ASCII to Morse code look-ups which is used by the "Ham Shack" program.

```
(00)=0020 00 00 00 00
(04)=0024 00 00 00 CE
(08)=0028 00 00 00 00
(0C)=002C CE 57 56 94
(10)=0030 F1 7C 3C 1C
(14)=0034 0C 04 04 04
(18)=0038 E4 F4 1C 3C
(1C)=003C 00 8C 00 32
(20)=0040 00 60 88 A8
(24)=0044 90 40 28 D0
(28)=0048 08 20 78 80
(2C)=004C 48 E0 A0 F0
(30)=0050 68 D8 50 10
(34)=0054 C0 30 18 70
(38)=0058 98 88 C8 00
```

```
>LIST
   0 REM   SPELUNKER  I
   1 GOTO 30000: REM  TO INITIALIZE
1000 PRINT "?";: INPUT IN$:IN$( LEN(IN$)+1)=" GO N * ":I=1
1005 NOUN=0:VERB=0
1010 GOSUB 1500: GOSUB 1600: GOSUB 1700
1020 IF W3$#"*    " THEN 1010
1050 NTYP=NTYP(NOUN):VTYP=VTYP(VERB)
1060 IF (VTYP MOD (NTYP*2))>=NTYP THEN 2000
1070 PRINT "ICH VERSTEHE NICHT"
1080 GOTO 3000
1200 GOTO 2000
1500 W3$="":S=0: FOR I=I TO LEN(IN$): IF S=0 THEN 1520: IF IN$(I,I)=" " THEN
     1580: IF S=5 THEN 1560: GOTO 1540
1520 IF IN$(I,I)=" " THEN 1560
1540 S=S+1:W3$(S)=IN$(I,I)
1560 NEXT I
1580 IF S<5 THEN W3$(S+1)=SPC$(S+1)
1590 RETURN
1600 IF NOUN#0 THEN RETURN : FOR J=1 TO NUMN: IF W3$#NOUNS$(J*5-4,J*5) THEN
     NEXT J: IF J>=NUMN THEN RETURN :NOUN=J:W2$=W3$
1610 RETURN
1700 IF VERB#0 THEN RETURN : FOR J=1 TO NUMV: IF W3$#VERBS$(J*5-4,J*5) THEN
     NEXT J: IF J>=NUMV THEN RETURN :VERB=J:W1$=W3$
1710 RETURN
2000 REM  MOVE
2010 NLOC=0
2020 IF NOUN>8 THEN 2200
2030 IF (NOUN MOD 2)=1 THEN 2100
2040 IF LOC#11 AND LOC#8 THEN 1070
2100 GOTO 2100+NOUN*10
2110 NLOC=N(LOC): GOTO 2190
2120 NLOC=0: GOTO 2190
2130 NLOC=E(LOC): GOTO 2190
2140 NLOC=15: IF LOC=8 THEN NLOC=8: GOTO 2190
2150 NLOC=S(LOC): GOTO 2190
2160 NLOC=22: IF LOC=8 THEN NLOC=8: GOTO 2190
2170 NLOC=W(LOC): GOTO 2190
2180 NLOC=12: IF LOC=8 THEN NLOC=8: GOTO 2190
2190 IF RAFT=1 THEN NLOC= ABS (NLOC)
2191 RAFT=0:PLOC=LOC
2192 IF NLOC>0 THEN LOC=NLOC
2193 IF NLOC#12 THEN 2900
2194 IF M(50)<5 THEN 2900
2195 IF PLOC=6 THEN S(12)=0
2196 IF PLOC=11 THEN W(12)=0
2197 GOTO 2900
2200 IF (NOUN=9 OR NOUN=10) AND ROPE=0 THEN GOTO 1070
2205 IF NOUN#9 THEN 2250
2210 IF LOC#5 AND LOC#13 THEN 1070
2220 IF LOC=5 THEN LOC=4
2230 IF LOC=13 THEN LOC=50
2240 GOTO 3000
2250 IF NOUN#10 THEN 2300
2260 IF LOC#4 THEN 1070
2270 LOC=5: GOTO 3000
2300 IF VERB=8 OR VERB=11 THEN 2320: GOTO 2350
2320 IF NUMP=8 THEN 1070
2325 IF NOUN=34 AND (LOC=19 OR LOC=10 OR LOC=5 OR LOC=18 OR LOC=7 OR LOC=
     6 OR LOC=11) THEN 2345
```

```
2330 IF STA(NOUN)#LOC THEN 1070
2335 IF NOUN=28 AND M(50)>0 THEN 1070
2345 STA(NOUN)=-1: GOTO 3000
2350 IF VERB=9 OR VERB=10 OR VERB=14 THEN 2370: GOTO 2400
2370 IF STA(NOUN)#-1 THEN 1070
2380 STA(NOUN)=LOC
2383 IF NOUN#33 THEN 2420
2385 IF VERB#10 THEN STA(33)=0
2387 LIGHT=0
2390 GOTO 2420
2400 IF VERB#12 THEN 2900
2410 IF STA(NOUN)#-1 THEN 1070
2420 FOR WT=1 TO NUMW*5-4 STEP 5
2425 IF NOUN#WT(WT) THEN 2480
2430 FOR D=2 TO 4
2435 IF (STA(WT(WT+D)) MOD 100)#LOC THEN 2470
2440 FOR M=1 TO NUMM*10-9 STEP 10
2445 IF WT(WT+D)#M(M) THEN 2460
2446 HT= RND (WT(WT+1))/(COURSE+1)
2448 M(M+9)=M(M+9)-HT
2449 IF M(M+4)=0 THEN M(M+4)=1
2450 PRINT "ASSAULT ON ";NOUNS$(M(M)*5-4,M(M)*5);", ";HT;" UNITS"
2452 PRINT "ITS LIFE FORCE IS NOW ";M(M+9);"%"
2455 IF M(M+9)>0 THEN 2460
2456 PRINT NOUNS$(M(M)*5-4,M(M)*5);" HAS BEEN ELIMINATED"
2457 STA(M(M))=0
2460 NEXT M
2470 NEXT D
2480 NEXT WT
2490 IF NOUN#16 OR VERB=10 THEN 2500
2492 STA(16)=0: GOTO 2493+ RND (4)
2493 N(LOC)=0: GOTO 2500
2494 E(LOC)=0: GOTO 2500
2495 S(LOC)=0: GOTO 2500
2496 W(LOC)=0
2500 IF NTYP#32 THEN 2900
2510 IF NOUN#33 THEN 2520: IF VERB=12 THEN LIGHT=1: GOTO 2900
2520 IF NOUN#29 THEN 2530:RAFT=1: GOTO 2900
2530 IF NOUN#30 THEN 2540:ROPE=1: GOTO 2900
2540 REM
2900 IF NOUN<11 THEN ROPE=0
2910 IF STA(30)=LOC THEN ROPE=1
2920 IF LOC=12 THEN 3000
2930 W(12)=6:S(12)=11
3000 REM  RE-ACTION
3010 FOR M=1 TO NUMM*10-9 STEP 10
3020 IF STA(M(M))#0 THEN GOSUB 3800
3030 NEXT M
3040 IF STA(35)=0 AND STA(34)=2 THEN STA(35)=2
3090 GOTO 4000
3800 REM  MONS SUB
3802 MRM=STA(M(M)) MOD 100
3810 IF (STA(M(M+1)) MOD 100)=MRM AND M(M+4)=0 THEN 3900
3820 IF MRM=LOC THEN 3860
3830 M(M+2)=(M(M+2)+M(M+3)) MOD 6
3840 IF M(M+2)#0 THEN RETURN
3845 GOTO 3850+ RND (4)
3850 NLOC=N(MRM): GOTO 3855
3851 NLOC=E(MRM): GOTO 3855
3852 NLOC=S(MRM): GOTO 3855
```

```
3853 NLOC=W(MRM): GOTO 3855
3855 IF NLOC<1 THEN RETURN
3858 STA(M(M))=NLOC+STA(M(M))-MRM: RETURN
3860 M(M+4)=M(M+4)+1
3865 KP=(M(M+5)-(STA(M(M+1))=-1)*40+9*(M(M+4)-2))*M(M+9)/100+CURSE
3866 IF KP>60 THEN KP=60
3870 SAP=(M(M+6)+9*(M(M+4)-2))*M(M+9)/100+CURSE
3871 IF SAP>70 THEN SAP=70
3875 SRP=(M(M+7)+9*(M(M+4)-2))*M(M+9)/100+CURSE
3876 IF SRP>80 THEN SRP=80
3877 PRINT "ATTACK BY ";NOUNS$((M(M)-1)*5+1,M(M)*5)
3879 R1= RND (100):R2= RND (100):R3= RND (100)
3880 IF KP>R1 THEN 3920
3885 IF SAP>R2 THEN 3940
3887 IF STA(M(M+1))#-1 THEN RETURN
3890 IF SRP>R3 THEN 3960
3895 RETURN
3900 STA(M(M))=M(M+8)
3905 STA(M(M+1))=M(M+8)
3910 RETURN
3920 VTAB 23: TAB 1: PRINT "THE ";NOUNS$((M(M)-1)*5+1,M(M)*5);" KILLED YOU!"

3924 PRINT KP,R1
3925 END
3940 FOR I=1 TO NUMN
3945 IF NTYP(I)=16 AND STA(I)=-1 THEN STA(I)=M(M+8)
3950 NEXT I
3957 PRINT "ALL YOUR REWARDS STOLEN"
3958 PRINT SAP,R2
3959 GOTO 3900
3960 PRINT "HE TOOK BACK HIS VALUABLE"
3962 PRINT SRP,R3
3965 GOTO 3900
4000 REM  OUTPUT
4020 FOR I=3 TO 9: VTAB I: TAB 2: PRINT "                              ": NEXT
     I
4050 IF LOC<1 OR LOC>50 THEN GOTO 4051
4060 GOTO 4000+100*LOC
4070 POKE 50,63: VTAB 3: TAB 2: PRINT LOC$;: POKE 50,255: PRINT "   ";: RETURN

4090 VTAB 23: TAB 1
4095 IF LIGHT=1 OR LOC<3 OR LOC=19 THEN 9100
4097 PRINT "IT IS VERY DARK"
4099 GOTO 9100
4100 LOC$="MOUTH       ": GOSUB 4070
4199 GOTO 4090
4200 LOC$="TREE ROOM   ": GOSUB 4070
4299 GOTO 4090
4300 LOC$="WRITING ROOM": GOSUB 4070
4399 GOTO 4090
4400 LOC$="PIT         ": GOSUB 4070
4499 GOTO 4090
4500 LOC$="SOUTH LAKE  ": GOSUB 4070
4599 GOTO 4090
4600 LOC$="WEST LAKE   ": GOSUB 4070
4699 GOTO 4090
4700 LOC$="NORTH LAKE  ": GOSUB 4070
4799 GOTO 4090
4800 LOC$="MAZE ROOM   ": GOSUB 4070
4899 GOTO 4090
```

```
4900 LOC$="FROZEN RIVER": GOSUB 4070
4999 GOTO 4090
5000 LOC$="RIVER ROOM  ": GOSUB 4070
5099 GOTO 4090
5100 LOC$="HUB ROOM    ": GOSUB 4070
5199 GOTO 4090
5200 LOC$="ICE ROOM    ": GOSUB 4070
5299 GOTO 4090
5300 LOC$="CHIMNEY     ": GOSUB 4070
5399 GOTO 4090
5400 LOC$="GOLD ROOM   ": GOSUB 4070
5499 GOTO 4090
5500 LOC$="BONES       ": GOSUB 4070
5510 IF STA(35)#-1 THEN CURSE=CURSE+15
5599 GOTO 4090
5600 LOC$="BATS        ": GOSUB 4070
5699 GOTO 4090
5700 LOC$="GHOST ROOM  ": GOSUB 4070
5799 GOTO 4090
5800 LOC$="MISTY LAKE  ": GOSUB 4070
5899 GOTO 4090
5900 LOC$="SWIFT RIVER ": GOSUB 4070
5999 GOTO 4090
6000 LOC$="INTERSECTION": GOSUB 4070
6099 GOTO 4090
6100 GOTO 6000
6200 GOTO 6000
6999 GOTO 4090
8900 LOC$="OVER FALLS  ": GOSUB 4070
8910 VTAB 23: TAB 1: GOTO 9090
9000 LOC$="YOUR HOME   ": GOSUB 4070
9005 AMT=0
9010 IF STA(25)=-1 THEN AMT=AMT+13
9020 IF STA(26)=-1 THEN AMT=AMT+22
9030 IF STA(27)=-1 THEN AMT=AMT+8
9040 IF STA(28)=-1 THEN AMT=AMT+5
9050 VTAB 23: TAB 1
9060 IF AMT=0 THEN 9090
9070 PRINT "YOU HAVE FOUND $";AMT;",";" AND (900)+100;" IN TREASURES"
9080 IF AMT>13 THEN PRINT "NICE SPELUNKING!"
9090 PRINT "GOOD-BYE"
9099 END
9100 FOR I=2 TO 10: VTAB I: TAB 30: PRINT "       ": NEXT I
9105 IF LIGHT=0 AND LOC>2 AND LOC#19 THEN 9290
9110 VTAB 5: TAB 33: PRINT "^": TAB 33: PRINT "+": POKE 50,63
9140 IF N(LOC)=0 OR (N(LOC)<0 AND RAFT=0) THEN 9150: VTAB 3: TAB 33: PRINT
     "N": TAB 33: PRINT " "
9150 IF S(LOC)=0 OR (S(LOC)<0 AND RAFT=0) THEN 9160: VTAB 8: TAB 33: PRINT
     " ": TAB 33: PRINT "S"
9160 IF E(LOC)=0 OR (E(LOC)<0 AND RAFT=0) THEN 9170: VTAB 6: TAB 35: PRINT
     " E"
9170 IF W(LOC)=0 OR (W(LOC)<0 AND RAFT=0) THEN 9180: VTAB 6: TAB 30: PRINT
     "W "
9180 IF (LOC=5 OR LOC=13) AND ROPE=1 THEN 9185: GOTO 9190
9185 VTAB 2: TAB 33: PRINT "UP"
9190 IF LOC#4 OR ROPE=0 THEN 9200
9195 VTAB 10: TAB 33: PRINT "DOWN"
9200 IF LOC=11 OR LOC=8 THEN 9210: GOTO 9290
9210 VTAB 3: TAB 30: PRINT "N ": TAB 30: PRINT " W"
9215 IF LOC#8 THEN 9220: VTAB 3: TAB 35: PRINT " E": TAB 35: PRINT "N "
```

```
 9220 VTAB 8: TAB 30: PRINT " W";: TAB 35: PRINT "S ": TAB 30: PRINT "S ";
      : TAB 35: PRINT " E"
 9290 POKE 50,255
 9300 IF LIGHT=0 AND LOC>2 AND LOC#19 THEN 9400
 9305 VTAB 5: TAB 2:J=0
 9310 FOR I=1 TO NUMN-1
 9320 IF (STA(I) MOD 100)#LOC THEN 9360
 9330 PRINT NOUNS$((I-1)*5+1,I*5);" ";
 9340 J=(J+1) MOD 4: IF J#0 THEN 9360
 9350 PRINT "": TAB 2
 9360 NEXT I
 9400 VTAB 13: TAB 2: FOR I=1 TO 12: PRINT "          ";: NEXT I
 9410 VTAB 13: TAB 2: PRINT "POSSESSIONS ";:NUMP=0
 9420 FOR I=1 TO NUMN-1
 9430 IF STA(I)>=0 THEN 9480
 9440 PRINT NOUNS$((I-1)*5+1,I*5);" ";
 9450 NUMP=NUMP+1: IF NUMP=4 THEN TAB 14
 9480 NEXT I
 9900 VTAB 23: TAB 1: GOTO 1000
30000 REM  INITIALIZE ROUTINE
30010 DIM IN$(40),NOUNS$(255),VERBS$(255),W1$(5),W2$(5),W3$(5),NTYP(50),VTYP(
      50),STA(50)
30020 DIM N(50),E(50),S(50),W(50)
30030 TEXT : CALL -936
30040 DIM LOC$(26),SPC$(5),POSS(10),M(6*10)
30050 SPC$="     "
30060 NUMW=6
30065 DIM WT(5*NUMW)
30070 LOC=1
30100 REM  INITIALIZE VARIABLES
30101 REM  SHOULD BE READ AND DATA STMTS
30110 NOUNS$( LEN(NOUNS$)+1)="N    NE   E    SE   S    SW   W    NW   UP   DOWN
       "
30120 NOUNS$( LEN(NOUNS$)+1)="CAVE LAKE RIVERTREE "
30130 NOUNS$( LEN(NOUNS$)+1)="AX   BOMB CURSEFIRE KNIFE"
30140 NOUNS$( LEN(NOUNS$)+1)="CLAM BATS BONESGHOSTOGRE "
30150 NOUNS$( LEN(NOUNS$)+1)="CHESTGOLD PEARLLAMP "
30160 NOUNS$( LEN(NOUNS$)+1)="RAFT ROPE TENT TRUCKLIGHT"
30170 NOUNS$( LEN(NOUNS$)+1)="WATERAPPLEICE  "
30195 NOUNS$( LEN(NOUNS$)+1)="*****"
30199 NUMN=37
30210 VERBS$( LEN(VERBS$)+1)="GO   JUMP RUN  WALK DRIVECLIMB"
30220 VERBS$( LEN(VERBS$)+1)="DIG  "
30230 VERBS$( LEN(VERBS$)+1)="CARRYDROP PUT  TAKE USE  WISH THROW"
30240 VERBS$( LEN(VERBS$)+1)="HELP KILL STOP HIT  FIGHT"
30250 VERBS$( LEN(VERBS$)+1)="RUB  "
30260 VERBS$( LEN(VERBS$)+1)="STARTDRIVE"
30270 VERBS$( LEN(VERBS$)+1)="DRINKEAT  BITE "
30295 VERBS$( LEN(VERBS$)+1)="*****"
30299 NUMV=26
30310 FOR I=1 TO 10:NTYP(I)=1: NEXT I
30320 FOR I=11 TO 14:NTYP(I)=2: NEXT I
30330 FOR I=15 TO 19:NTYP(I)=4: NEXT I
30340 FOR I=20 TO 24:NTYP(I)=8: NEXT I
30350 FOR I=25 TO 28:NTYP(I)=16: NEXT I
30360 FOR I=29 TO 33:NTYP(I)=32: NEXT I
30370 FOR I=34 TO 35:NTYP(I)=64: NEXT I
30380 NTYP(36)=32
30410 FOR I=1 TO 6:VTYP(I)=1: NEXT I
30412 VTYP(2)=11:VTYP(6)=3
```

```
30420 VTYP(7)=2
30430 FOR I=8 TO 11:VTYP(I)=116: NEXT I
30432 VTYP(12)=36:VTYP(13)=36:VTYP(14)=4
30440 FOR I=15 TO 19:VTYP(I)=8: NEXT I
30442 VTYP(17)=40
30450 VTYP(20)=16
30460 FOR I=21 TO 22:VTYP(I)=32: NEXT I
30470 FOR I=23 TO 25:VTYP(I)=64: NEXT I
30500 FOR I=1 TO 14:STA(I)=0: NEXT I
30510 STA(15)=4:STA(16)=3:STA(17)=15
30520 STA(18)=13:STA(19)=1:STA(20)=18
30530 STA(21)=16:STA(22)=15:STA(23)=17
30540 STA(24)=14:STA(25)=16:STA(26)=14
30550 STA(27)=18:STA(28)=12:STA(29)=5
30560 STA(30)=9:STA(31)=1:STA(32)=1
30570 STA(33)=1:STA(34)=0:STA(35)=0
30580 STA(36)=12
30600 FOR I=1 TO 50:N(I)=0:E(I)=0:S(I)=0:W(I)=0: NEXT I
30610 N(1)=50:N(3)=2:N(5)=-18:N(7)=9:N(8)=8:N(9)=7
30620 N(10)=3:N(11)=13:N(16)=22:N(18)=7
30630 N(19)=1:N(20)=8:N(22)=11
30640 E(2)=1:E(4)=20:E(6)=-18:E(8)=9:E(9)=1:E(11)=14:E(17)=21:E(20)=3
30650 E(21)=11
30660 S(1)=19:S(2)=3:S(3)=10:S(7)=-18:S(8)=8:S(10)=-11:S(11)=-49:S(12)=11:
      S(13)=11:S(18)=5
30670 S(21)=22:S(22)=16
30680 W(1)=2:W(3)=20:W(6)=12:W(8)=20:W(9)=8:W(11)=21
30690 W(12)=6:W(14)=11:W(15)=11:W(18)=6:W(20)=4:W(21)=17:W(22)=21
30700 POKE 50,63
30710 VTAB 24: GOSUB 31999: VTAB 1: GOSUB 31999: VTAB 11: GOSUB 31999: VTAB
      16: GOSUB 31999
30720 VTAB 2: TAB 1
30730 FOR I=2 TO 23: PRINT " ";: TAB 29: IF I<11 THEN PRINT " ";: TAB 39: PRINT
      " ": NEXT I
30740 POKE 50,255: POKE 32,1: POKE 33,37: POKE 34,16: POKE 35,23: VTAB 17:
       TAB 2
30800 FOR I=1 TO 60:M(I)=0: NEXT I
30810 M(1)=24:M(2)=26:M(4)=2:M(6)=60:M(7)=30:M(8)=55:M(9)=14:M(10)=100
30820 M(11)=21:M(12)=25:M(14)=4:M(16)=60:M(17)=40:M(18)=90:M(19)=16:M(20)=
      40
30830 M(21)=23:M(24)=6:M(29)=17:M(30)=50
30840 M(31)=20:M(32)=27:M(33)=1:M(36)=90:M(37)=60:M(38)=65:M(39)=18:M(40)=
      60
30850 M(41)=36:M(42)=28:M(43)=1:M(47)=60:M(49)=12:M(50)=25
30860 M(51)=22:M(53)=1:M(59)=15:M(60)=75
30890 NUMM=6
30900 WT(1)=15:WT(2)=100:WT(3)=24:WT(4)=0:WT(5)=0
30910 WT(6)=16:WT(7)=150:WT(8)=24:WT(9)=22:WT(10)=36
30920 WT(11)=18:WT(12)=30:WT(13)=21:WT(14)=22:WT(15)=36
30930 WT(16)=19:WT(17)=50:WT(18)=24:WT(19)=20:WT(20)=0
30940 WT(21)=33:WT(22)=30:WT(23)=23:WT(24)=0:WT(25)=0
30950 WT(26)=36:WT(27)=40:WT(28)=21:WT(29)=0:WT(30)=0
30999 GOTO 4000
31999 TAB 1: PRINT "                                                ";: RETURN
32000 PRINT ( PEEK (202)+ PEEK (203)*256)-( PEEK (204)+ PEEK (205)*256): END
```

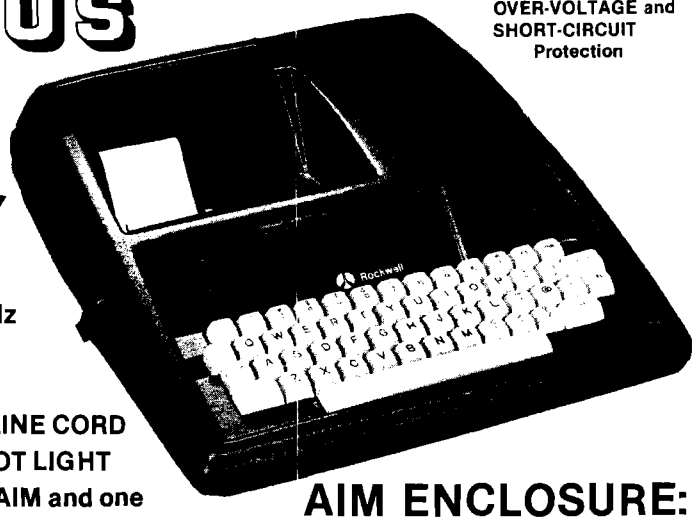
The COMPUTERIST Has It All !!

The leaders in Expansion Accessories for
AIM / SYM / KIM

Featuring a Power Supply/Enclosure for the AIM 65

AIM PLUS™

Now with -
OVER-VOLTAGE and
SHORT-CIRCUIT
Protection



ENCLOSURE
WITH BUILT IN
POWER SUPPLY

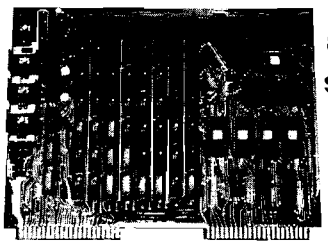
SPECIFICATIONS:
INPUT: 110/220 VAC 50/60 Hz
OUTPUT: +5V @ 5A
+24V @ 1A
GROUNDED THREE-WIRE LINE CORD
ON/OFF SWITCH WITH PILOT LIGHT
Enclosure has room for the AIM and one additional board: MEMORY PLUS or VIDEO PLUS

AIM ENCLOSURE: $45.00

AIM PLUS: $110.00 **AIM and AIM PLUS: $485.00**

Plus some very elegant expansion boards

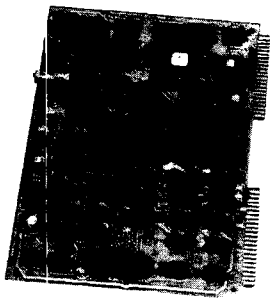
MEMORY PLUS™ FOR AIM/SYM/KIM



8K STATIC RAM LOW POWER
Sockets for 8K Eprom
6522 1/0 Port
ON BOARD REGULATORS
EPROM PROGRAMMER

MEMORY PLUS: $200.00 FULLY ASSEMBLED AND TESTED

VIDEO PLUS™ FOR AIM/SYM/KIM



UPPER/lower case ASCII
128 Additional User Programmable Characters: GRAPHICS-SYMBOLS-FOREIGN CHARACTERS
Programmable Screen Format up to 80 CHARACTERS - 24 LINES
KEYBOARD and LIGHT PEN Interfaces
Up to 4K DISPLAY RAM
Provision for 2K EPROM
Provision to add 6502 for STAND-ALONE SYSTEM
ASSEMBLED AND TESTED WITH 2K DISPLAY RAM

VIDEO PLUS: $245.00

Many other products available. Write for a complete catalog [or call if you are in a hurry !]

The prices listed above do not include shipping and handling.

THE COMPUTERIST INC

PO Box 3
S Chelmsford, MA 01824
617/256-3649

Classified Ads

CHALLENGER IP OWNERS: Tired of 24X24 graphics? $40 worth of components will upgrade your video display to give 30 lines X 50 characters on screen and double your processor speed. Complete software and step by step hardware modifications for only **$8.00.** Contact:
S. Chalfin
905 Clinton St.
Philadelphia, PA 19107

PET LANGUAGE MACHINE GUIDE comprehensive manual to aid the machine language programmer. More than 30 routines are fully detailed so that the reader can put them to use immediately.Specify old or new ROMS. $6.95 plus .75 for postage, Visa or Mastercharge accepted. Money back guarantee (10 days). Contact:
ABACUS SOFTWARE
P.O. Box 7211
Grand Rapids, MI 49510

WANTED Disk operating system, hardware and software for use with 16K SYM-1. Contact:
Kenneth A. Edelstein
341 West 71st Street
New York, NY 10023

SYM-1 OWNERS–SYM/KIM Appendix to First Book of KIM details changes to KIM games run on BASIC SYM-1. Changes shown line by line; load, modify, and run. Appendix only $4. First Book of KIM $9., combo both $12.50 postpaid. Order from:
Robert A. Peck
P.O.Box 2231
Sunnyvale, CA 94087

For only $33 for 12 issues you get CURSOR, the original magazine for the Commodore PET. Each issue has a graphic "Front Cover" program plus five excellent programs that are ready to Load and Run. The CURSOR NOTES newsletter with each issue provides useful information about the programs, as well as help on using your PET. Contact:
CURSOR
Box 550
Goleta, CA 93017

SUPER 8-K HORSERACE: Up to ten people can go to the track and bet on 8 different horses each race. Every race is new, with graphics and sound. Best game of this type. On cassette for an 8K APPLE II for only $9.95. Contact:
Tim Dowling
7243 Eastview Drive
Tucson, AZ 85710

APPLE, PET ATARI salespersons needed throughout the world. Work for commissions. Send resume to Recruiting Dept., COMPUTERWORLD Inc.,
c/o Computer Components of Orange County
6791 Westminster Ave.
Westminster, CA 92683

OPTIMIZE APPLESOFT programs: shorten variable names, remove remarks and extra colons, concatenate lines, renumber, list variable cross-references. (2) 1.3K programs for (16-48K) APPLE II's. Casette $15, disk $20 from:
Sensible Software
P.O. Box 2395
Dearborn, MI 48123

NEW BOWLING PROGRAM FOR APPLE: One to four players compete in a standard bowling game. Challenging game with excellent quality APPLESOFT program. Cassette $9.95. Order from:
C.E.Howerton
125 Marcella Road
Hampton, VA 23666

"STAR TREK" for your OSI Challenger IP or Superboard II, needs 5K RAM. Runs in machine language. Very fast and tough to beat. Send check or money order for $12.95 to:
David Van't Hof
RR1
Hospers, IA 51238

Advertising Software for APPLE:
$25 buys SCROLLING WONDER, GIANT LETTER, and HI-RES ALPHANUMERICS: on cassette, 16kb. $25 buys MULTI-MESSAGE with INTERLEAVED KALEIDOSCOPE, and MULTI-MESSAGE with INTERLEAVED ABSTRACT ART: on cassette, 32kb.
Send check or money order to:
CONNECTICUT INFORMATION SYSTEMS
218 Huntington Rd.
Bridgeport, CT 06608

QUALITY SOFTWARE for the PET: Cheque-Check (best balancer) $7.95; Metric-Calc (RPN, with conversions) $7.95; Mem-Explorer (learn PET details) $7.95; Billboard ("Times Square" message of choice) $49.95; Send check or money order to:
MICRO SOFTWARE SYSTEMS
Dept. M879, P.O. Box 1442
Woodbridge, VA 22193
VA res. add 4% tax.

MICRO

6522 Timing and Counting Techniques

While many 6502 computerists are becoming familar with the 6522 Versatile Interface Adapter, do you really know how all of it features work or how to use them? This tutorial will clear up the mysteries of the 6522.

Marvin L. De Jong
Department of Math and Physics
The School of the Ozarks
Point Lookout, MO 65726

Applications that require interval timers include everything from the production of simple sound effects for games to the implementation of sophisticated data logging or control processes. Because single-chip microcomputers, such as the Rockwell 6500/1 and the Intel 8048, are intended for high volume, low cost applications, the fact that they include counter/timer logic is a testimony to the importance of counter/timer functions for a large variety of applications. Several simple applications will be explained.

The techniques will focus on the counter/timers found on the 6522 Versatile Interface Adapter. The 6522 is currently popular in a number of microcomputer systems that utilize the 6502, including the SYM-1, the AIM 65, and the MICRO PLUS. Expansion boards such as the MEMORY PLUS also include the 6522, and the 6522 can be easily interfaced to the popular KIM-1 (see *6502 User Notes*, No. 13, pg. 16). However, the techniques that are described will frequently be applicable to any counter/timer with only minor modifications in the hardware or the programs.

The basic features included in many counter/timers (also called interval timers) are shown in Figure 1. This block diagram shows that a counter/timer consists of three registers; the counter register which is either an 8-bit register or a 16-bit register, a flag register, and a control register. A number, N, is loaded into the counter register by a WRITE (typically an STA) instruction. If the counter is a 16-bit register, then two write instructions are required. In 6502

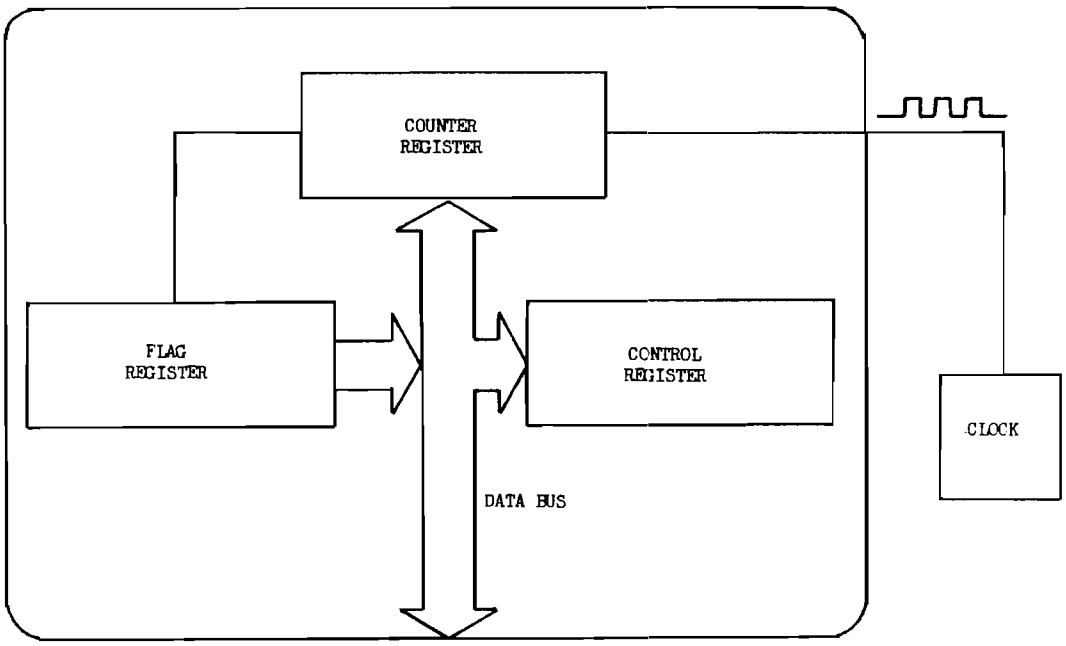


Figure 1. Block Diagram of a Typical Counter/Timer.

systems these registers are simply some of the 65536 memory locations. After N is loaded into the counter, it is decremented at a rate determined by the clock signal connected to the counter.

When N decrements *through* zero, one of the bits in the flag register is set to logic one. Thus, the contents of the counter register change as follows: N, N-1, N-2, ..., 2, 1, 0, and on the next clock cycle the flag is set. Consequently it actually takes N + 1 clock cycles to "time out." This summarizes the fundamentals of the counting/timing process.

The control register is used to select one of several modes available to the programmer. For example, in one mode the contents of the counter register are decremented at the same rate as the system clock, while in another mode pulses on an external pin cause the counter to decrement, and in a third mode the counter is automatically reloaded after each time-out. The modes available with a 6522 will be discussed in more detail below.

The 6522 Interval Timers

The 6522 Versatile Interface Adapter is a complex integrated circuit that includes two eight-bit I/O ports, four pins associated with handshaking signals for these two I/O ports, and two interval timers. The I/O ports and handshaking pins will only be of incidental interest, and we will describe the use of a few of these features as the need arises. Our principal interest is in the two counter/timers that are available on the 6522, called T1 and T2 respectively. Of course, the various registers needed to detect timing-out and to select the various timing modes will also be of interest.

In most 6502 microcomputer systems, the 6522 will be interfaced to occupy 16 contiguous memory locations. The AIM 65 and SYM-1, for example, use locations with addresses $A000 to $A00F for the 6522. Table I summarizes the names of each of these 16 locations, while Table II lists the functions of the registers. Of particular interest are the timer locations $A004 through $A009, the interrupt flag register (IFR), and the control register (ACR). These correspond precisely with the registers mentioned above in connection with Figure 1. That is, the IFR is the flag register and the ACR is the control register.

Both counter/timers, T1 and T2, on the 6522 are 16-bit devices; that is, a 16-bit number is loaded into the counter register and then decremented until time-out. Because the counter registers are 16-bit registers, two WRITE operations are needed to load the counter since only eight bits of data can be written at one time.

To prevent one eight-bit number (the low-order byte) from being decremented

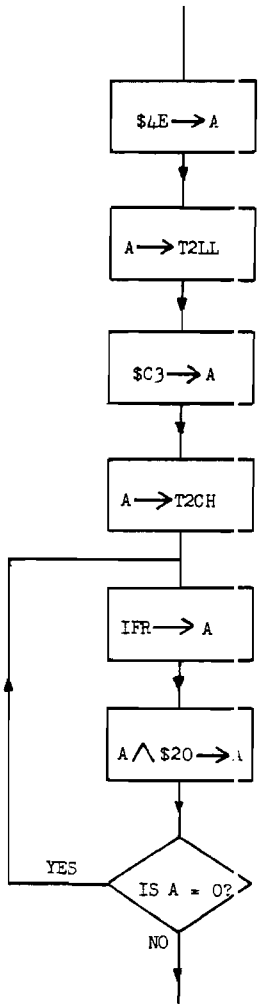


Figure 2. Flowchart of a Simple Interval Timer Delay Loop.

while the other (the high-order byte) is still not loaded, temporary storage *latches* are provided. Using the T2 timer as an example, the low-order eight bits of the number, N, to be loaded into the counter are loaded into the low-order byte of the T2 latch (T2LL). Nothing happens. Next, the high-order eight bits of N are loaded into the high-order byte of the T2 counter. Referring to Table II, this last operation has three important and simultaneous consequences:

- The byte stored in the T2 latch (T2LL) is transferred to the low-order byte of the T2 counter (T2CL). T2 now contains a 16-bit number.
- The interrupt flag that signals the time-out, bit five of the IFR, is cleared (set to zero). It will be set (to one) when the number N decrements through zero.
- The countdown begins.

The T1 timer has two latches, one to store the low-order byte to be transferred to the counter, and one to store the high-order byte to be transferred to the counter. One reason for this difference is that the T1 timer has a "free-running" mode. At the end of one time-out, the two bytes of data stored in the latches are *automatically* transferred to the 16-bit T1 counter to start a new timing interval.

Furthermore, the values in the two latches may be changed during one timing interval to give a new value for the next interval. The examples that follow should make these points clear. Additional discussion of the READ operations outlined in Table II will also be posponed until required by a specific example.

A Simple Delay Loop Using the T2 Timer

The most common application of counter/timers is the implementation of delay loops. The counter/timer replaces a series of instructions that are designed to waste time. The counter/timer simplifies greatly the instructions that are necessary to program a time delay, and furthermore, the computer may execute other tasks during the delay produced by the timer, a feat that is much more difficult to perform with a software implemented delay loop.

An assembly language version of a simple delay loop using the T2 timer on the 6522 is listed in Table III. The mnemonics are perfectly general for 6502 systems, but the addresses of the registers of the 6522 are the ones given in Table II for the AIM 65 and the SYM-1. Programmers using other systems need only change the addresses to correspond to the locations of the 6522 registers in the address space of their

Table I. Memory Assignment Names for the 6522 VIA.

ADDRESS	SYMBOL	NAME
$A000	ORB	Port B Input/Output Registers
$A001	ORA	Port A Input/Output Registers (with handshaking)
$A002	DDRB	Port B Data Direction Register
$A003	DDRA	Port A Data Direction Register
$A004	T1LL	Timer 1 Latch Low-order Byte (WRITE)
$A004	T1CL	Timer 1 Counter Low-order Byte (READ)
$A005	T1LH	Timer 1 Latch High-order Byte (WRITE)
$A005	T1CH	Timer 1 Counter High-order Byte (READ)
$A006	T1LL	Timer 1 Latch Low-order Byte (READ or WRITE)
$A007	T1LH	Timer 1 Latch High-order Byte (READ or WRITE)
$A008	T2LL	Timer 2 Latch Low-order Byte (WRITE)
$A008	T2CL	Timer 2 Counter Low-order Byte (READ)
$A009	T2CH	Timer 2 Counter High-order Byte (READ or WRITE)
$A00A	SR	Shift Register
$A00B	ACR	Auxiliary Control Register (Control Register for Timers)
$A00C	PCR	Peripheral Control Register
$A00D	IFR	Interrupt Flag Register (Status Register)
$A00E	IER	Interrupt Enable Register
$A00F	ORA	Port A I/O Register (without handshaking)

systems. Pay careful attention to the comments in Table III, because they relate each step to points in our previous discussion. Figure 2 is a flowchart of the delay loop, and it has a box for each of the instructions in Table III.

In the program listing given in Table III, timing begins at the completion of the STA T2CH instruction. The program waits in the loop consisting of the series of instructions LDA IFR, AND $20, BEQ WAIT until the time-out of the T2 timer sets bit five of the interrupt flag register. The formula for the time T required for the interval timer to time-out is:

$$T = (N + 1)T_C$$

where N is the 16-bit number loaded into the counter and T_C is the clock period (typically one microsecond).

If the branch instructions (LDA IFR, AND $20, BEQ WAIT) are taken into account, then the total loop time, T_L, is given by the expression:

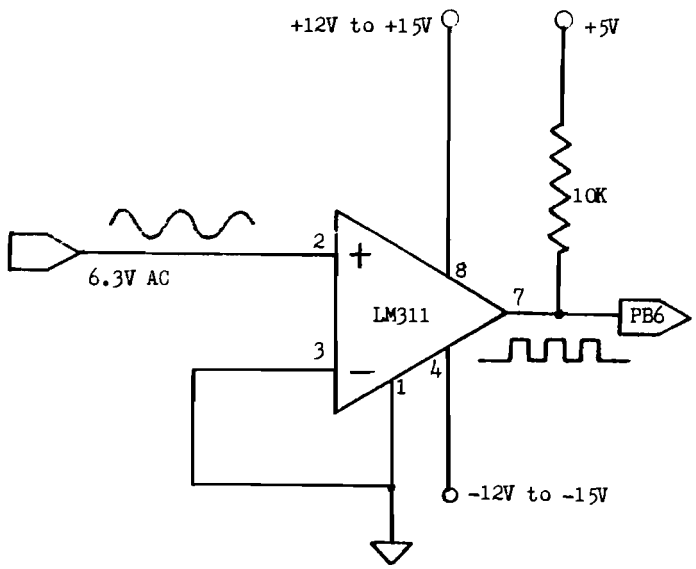
$$(N + 6)T_C \leq T_L \leq (N + 14)T_C$$

The uncertainty of eight cycles in the loop time arises from the uncertainty of where the T2 counter/timer actually times out in the series of test and branch instructions within the loop. For the numbers that were used in Table III,

Table II. Memory Assignments and Functions of Some of the Registers of the 6522 VIA.

ADDRESS	SYMBOL	FUNCTION
$A004	T1LL	WRITE (STA T1LL): Load an eight-bit number into the low-order byte of the T1 latch.
$A004	T1CL	READ (LDA T1CL): Read the contents of the low-order byte of the T1 counter, and clear the interrupt flag, bit six of the IFR.
$A005	T1LH	WRITE (STA T1LH): Load an eight-bit number into the high-order byte of the T1 latch, transfer the contents of both T1 latches to the T1 counters, clear the T1 interrupt flag, and start the counting process.
$A005	T1CH	READ (LDA T1CH): Read the contents of the high-order byte of the T1 counter.
$A006	T1LL	WRITE (STA T1LL): Load an eight-bit number into the low-order byte of the T1 latch.
$A006	T1LL	READ (LDA T1LL): Read the contents of the low-order byte of the T1 latch.
$A007	T1LH	WRITE (STA T1LH): Load an eight-bit number into the high-order byte of the T1 latch and clear the T1 interrupt flag.
$A007	T1LH	READ (LDA T1LH): Read the contents of the high-order byte of the T1 latch.
$A008	T2LL	WRITE (STA T2LL): Load an eight-bit number into the low-order byte of the T2 latch.
$A008	T2CL	READ (LDA T2CL): Read the contents of the low-order byte of the T2 counter, and clear the interrupt flag, bit five of the IFR.
$A009	T2CH	WRITE (STA T2CH): Load and eight-bit number into the high-order byte of the T2 counter, transfer the contents of the low-order byte in the T2 latch to the low-order byte of the T2 counter, clear the T2 interrupt flag, and start the counting process.
$A009	T2CH	READ (LDA T2CH): Read the contents of the high-order byte of the T2 counter.
$A00B	ACR	Bits five, six, and seven control the modes of T1 and T2.
$A00D	IFR	Bit six equal to one signals a time-out of the T1 counter/timer. Bit five equal to one signals a time-out of the T2 counter/timer.

Figure 3. 60 Hz Signal Conditioner for the Low Overhead Clock. A circuit based on the 555 timer and using only the +5V supply can be found in Berlin's 555 Timer Applications Sourcebook, pgs.2-13.



$T = (\$C34E + 1)T_C = 0.05$ seconds for a one microsecond clock. The loop time is between 5 and 13 microseconds longer. For many applications, this uncertainty will be of no consequence.

As pointed out earlier, the microprocessor need not be idle while the timer is timing out. For the particular delay of 0.05 seconds programmed in Table III, a total of 50,000 clock cycles elapse while the timer is running. During that time, between 25,000 and 10,000 instructions could be executed by the 6502. These instructions would be placed between the STA T2CH and the LDA IFR instructions. This is the principal advantage of the counter/timer implemented delay loop; that is, the microprocessor can be performing meaningful tasks during the timing-out process.

Counting Pulses — A 24-Hour Clock

The T2 timer can also be used to count pulses from an external source. This is useful for frequency counting (*MICRO*, June 1979, pg. 41) or any other event counting application such as radioactive half-life measurements. The T2 timer is placed in its pulse counting mode by setting bit five in the auxiliary control register (ACR) to logic one, and applying the TTL level pulses to bit six of port B, PB6. To illustrate this mode, and to illustrate how the timers can be used to generate interrupt requests (IRQs), we have chosen to describe a simple 24-hour clock that requires very little computer time overhead.

The 60 Hz power line frequency is sufficiently stable over long periods for many clocks. Somewhere in your microcomputer system you will probably be able to locate a low-voltage 60 Hz source. This is conditioned by the circuit shown in Figure 3 to produce a 60 Hz square wave, and the output is applied to PB6 to be counted. Clearly there are 3600 ($0E10) such pulses in a minute.

The T2 counter/timer will be programmed to count 3600 pulses followed by an interrupt request. The interrupt routine increments one location in memory to keep track of minutes, and when this location reaches 60, another location is incremented to keep track of the hours. At the beginning of the interrupt routine the T2 counter/timer is reloaded with 3600 for the next period.

The program is listed in Table IV. The first two instructions set bit five of the ACR to logic one. Next the timer is loaded with $0E0F. Note that $0E0F + 1 = 3600. The LDA $A0 and STA IER instructions enable interrupts from bit five of the interrupt flag register (IFR) of the 6522 to the 6502 microprocessor's IRQ pin, a connection that is usually internal to the microcomputer system.

To enable interrupt request signals from T2, bit five of the IER (interrupt enable register) must be set to logic one, with bit seven of the IER also set to logic one. At the end of the timing interval, not only will bit five of the IFR be set to one, but also the IRQ pin on the 6502 microprocessor will be pulled to logic zero, producing an interrupt request.

The next instruction after enabling the interrupt from the T2 timer is the CLI instruction that allows the 6502 to recognize these interrupts. The last instruction in the main program should not be taken literally. It is simply an infnite loop that represents the user's main program, a FORTRAN interpreter for example.

The interrupt routine is also given in Table IV. Timekeeping routines have been described in several other articles (*MICRO*, March 1979, pg. 5), so the details will not be repeated here. Note that in order for the program to execute, the IRQ vector must point to the starting address of the interrupt request routine, in our case $0300. Note also, that the program could be easily modified to keep track of seconds by counting only 60 pulses, something that can be done with an eight-bit counter like the one on the R650/1. The hours-minutes clock requires only about 50 microseconds per

Table III. A Simple Delay Loop Using the T2 Timer on the 6522.

```
                                .                       .
                                .                       .
                                .                       .
$0300  A9 4E          START     LDA $4E     Load the byte for the T2 latch low, then
$0302  8D 08 A0                STA T2LL    transfer it into T2 latch low (T2LL).
$0305  A9 C3                   LDA $C3     Load the byte for the T2 counter high,
$0307  8D 09 A0                STA T2CH    then transfer it into T2 counter high (T2CH)
$030A  AD 0D A0       WAIT      LDA IFR     Read the flag register, IFR.  Mask all bits
$030D  29 20                   AND $20     except bit five.  Check to see if bit five
$030F  F0 F9                   BEQ WAIT    is set.  No, then wait.  Yes, loop is finish
                                .                       .
                                .                       .
                                .                       .
```

minute of computing time, truly a low-overhead clock.

To display the minutes and hours, the user must provide a display routine that takes the contents of locations $0000 and $0001 and displays these numbers. Such a routine is not included in Table IV since the instructions used will depend on the microcomputer system, and previously written clock programs have included suitable display routines.

To summarize the operation of the T2 counter/timer on the 6522 we conclude this section with the following statements:

- To decrement the 16-bit number in the T2 counter at the system clock rate, *clear* bit five of the ACR.
- To decrement the 16-bit number in the T2 counter using external pulses applied to PB6 (pin 6 of Port B), *set* bit five of the ACR.
- To produce an interrupt request (IRQ) when the counter decrements through zero in either of its modes, *set* bits five and seven of the IER.
- To disable the interrupt feature, *set* bit five of the IER and clear bit *seven* of the IER.
- A system RESET disables the pulse-counting mode and the interrupt request feature by clearing all the registers of the 6522.

Table IV. Low Overhead 24-hour Clock.

```
$0200 A9 20      MAIN   LDA $20     Put T2 in its pulse-counting  mode
$0202 8D 0B A0          STA ACR      by setting bit five to logic one.
$0205 A9 0F             LDA $0F     Set up T2 to count 3600 pulses.
$0207 8D 08 A0          STA T2LL
$020A A9 0E             LDA $0E
$020C 8D 09 A0          STA T2CH
$020F A9 A0             LDA $A0     Set up interrupt enable register
$0211 8D 0E A0          STA IER      to permit IRQ from T2.
$0214 58                CLI         Allow 6502 to accept IRQ signals.
$0215 4C 15 02   HERE   JMP HERE    Loop here between interrupts.

                        INTERRUPT ROUTINE

$0300 A9 0E             LDA $0E     Start counting pulses again by
$0302 8D 09 A0          STA T2CH     loading T2CH.
$0305 18                CLC         Clear carry for addition.
$0306 F8                SED         Set decimal mode for addition.
$0307 A5 00             LDA MIN     Get minutes.
$0309 69 01             ADC $01     Add one.
$030B 85 00             STA MIN
$030D C9 60             CMP $60     Is one hour complete?
$030F D0 13             BNE DONE    No, get out of interrupt routine.
$0311 A9 00             LDA $00     Yes, set minutes to zero.
$0313 85 00             STA MIN
$0315 18                CLC
$0316 A5 01             LDA HRS     Get hours.
$0318 69 01             ADC $01     Add one.
$031A 85 01             STA HRS
$031C C9 24             CMP $24     Is one day complete?
$031E D0 04             BNE DONE
$0320 A9 00             LDA $00     Clear hours.
$0322 85 01             STA HRS
$0324 D8         DONE   CLD         Clear decimal mode.
$0325 40                RTI         Return to the main program.
```

Producing Long Time Delays

The maximum time delay that can be produced with the T2 counter/timer when it is decrementing at the system clock rate is approximately ($FFFF + 1)$T_C$ or 0.065536 seconds if T_C = 1 microsecond. In certain applications longer time delays are necessary. To obtain these delays, the T1 timer is used in conjunction with the T2 counter/timer. We digress for a moment to introduce the T1 timer.

The T1 timer can be used to implement a simple delay loop in exactly the same way as the T2 timer. Refer to Table III. If the addresses $A004 and $A005 replace addresses $A008 and $A009, respectively, and if bit six of the interrupt flag register (IFR) is tested rather than bit five, then the program in Table III will work in exactly the same way except that the T1 timer is being used.

The same equation gives the loop time and, as in the case of the T2 timer, the maximum delay is about 0.065 seconds. The T1 timer cannot, however, count pulses. Consequently it cannot replace the T2 timer in the program listed in Table IV. In place of the pulse counting mode, the T1 timer has a free-running mode, and it is capable of toggling the logic level on pin seven of Port B, PB7.

The initialization of the free-running mode with PB7 toggling is illustrated in a simple program shown in Table V. This program will produce a square wave output on PB7. The period of the square wave is given by the equation:

$$T_p = 2(N + 2)T_c$$

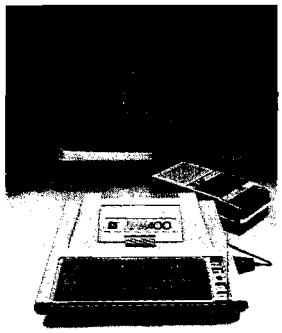
where T_p is the period of the square wave, N is the 16-bit number loaded into the T1 timer, and T_c is the period of the system clock (Typically one microsecond). The frequency of the square wave is $f = 1/T_p$.

To initialize this mode, bits seven and six of the auxiliary control register (ACR) must be set. Thus, the program in Table V begins by loading $C0 into the ACR. Timing is initiated by loading the high-order byte of N into location $A005 which corresponds to T1LH. Once started, the square wave will run forever, no matter what else is happening in the program, provided the registers that control the behavior of the T1 timer are not changed. That is, after the timer "times out", it will automatically reload the two counter registers from the numbers stored in its latches, T1LL and T1LH.

The last instruction in Table V is an infinite loop that simulates the user's program intended to run concurrently with generation of the square wave. Table VI lists some values for N that are frequently used in timing applications. If you have an oscilloscope, run the program with various values of N and connect the

(Continued on page 34)

ATARI® PERSONAL COMPUTER
CCI COMPUTER SYSTEMS DELIVERS . . . NOW!!!



ATARI® 400™
PERSONAL COMPUTER SYSTEM

PRICE INCLUDES:
Computer Console
BASIC Language Cartridge
BASIC Language Programming Manual (Wiley)
400 Operator's Manual with Note Book
Power Supply
TV Switch Box

$549.99

ATARI® 800™
PERSONAL COMPUTER SYSTEM

PRICE INCLUDES:
Computer Console
BASIC Language Cartridge
Education System Master Cartridge
BASIC Language Programming Manual (Wiley)
800 Operator's Manual with Note Book
ATARI 410 Program Recorder
Guide to BASIC Programming Cassette
8K RAM Module•Power Supply•
TV Switch Box

$999.99

WE PROMISE TO DELIVER!!!

- We GUARANTEE ship dates on prepaid Computer System orders. Send your orders in now!!!
- If for reasons beyond our control we miss a ship date, we will REFUND the shipping and handling charges and give you a 10% DISCOUNT on any future software purchases for your ATARI® System.*
- For prepaid system orders, get a FREE Accessory Controller of your choice.

*Order must be prepaid by Cashier's Check made out to Computer Components of Orange County.

ATARI USER'S GROUP INFORMATION . . . CALL (714) 891-2584

PERIPHERALS AND ACCESSORIES

ATARI® 810™ DISC DRIVE* $749.99

DISKETTES
CX8100 BLANK DISKETTES*
CX8101 DISK FILE MANAGER* $5.00/ea.

ATARI® 820™ PRINTER* $599.99

ACCESSORY CONTROLLERS
CX20-01 DRIVING CONTROLLER PAIR
CX30-04 PADDLE CONTROLLER PAIR
CX40-04 JOYSTICK CONTROLLER PAIR
$19.95/ea.

ATARI® 410™ PROGRAM RECORDER $89.99

ADD-ON MEMORY (800 ONLY)
CX852 8K RAM MEMORY MODULE $124.99
CX853 16K RAM MEMORY MODULE $249.99

SOFTWARE

ROM CARTRIDGES
CXL4001 EDUCATION SYSTEM MASTER CARTRIDGE $34.99
KEY: (j) = uses joystick controller
(p) = uses paddle controller
(d) = uses driving controller

GAMES . . . . . $49.95/ea.
CXL4004 BASKETBALL . . . . . (j)
CXL4005 LIFE . . . . . (j)
CXL4006 SUPER BREAKOUT™ . . . . . (p)
CX4008 SUPER BUG™** . . . . . (d)

APPLICATION $69.99
CXL4002 ATARI BASIC
CXL4003 ASSEMBLER DEBUG**
CXL4007 MUSIC COMPOSER
CXL4009 COMPUTER CHESS** . . . . . (j)

EDUCATION SYSTEM CASSETTE PROGRAMS $39.99/ea.

CX6001 U.S. HISTORY
CX6002 U.S. GOVERNMENT
CX6003 SUPERVISORY SKILLS
CX6004 WORLD HISTORY (WESTERN)
CX6005 BASIC SOCIOLOGY
CX6006 COUNSELING PROCEDURES
CX6007 PRINCIPLES OF ACCOUNTING
CX6008 PHYSICS
CX6009 GREAT CLASSICS (ENGLISH)
CX6010 BUSINESS COMMUNICATIONS
CX6011 BASIC PSYCHOLOGY
CX6012 EFFECTIVE WRITING
CX6013 AUTO MECHANICS
CX6014 PRINCIPLES OF ECONOMICS
CX6015 SPELLING
CX6016 BASIC ELECTRICITY
CX6017 BASIC ALGEBRA

BASIC GAME AND PROGRAM CASSETTES
CX4101 GUIDE TO BASIC PROGRAMMING*
CX4102 BASIC GAME PROGRAMS*
$29.95/ea.

*October Delivery **November Delivery

—Prices subject to change.—

COMPUTER COMPONENTS OF ORANGE COUNTY

6791 Westminster Ave., Westminster, CA 92683 714-891-2584 Telex 182274
Hours: Tues-Fri 11:00 AM to 8:00 PM—Sat 10:00 AM to 6:00 PM—Sun 12:00 to 4:00 PM (Closed Mon)
Master Charge, Visa, B of A are accepted. Allow 2 weeks for personal check to clear.
Add $2.00 for handling and postage. For computer systems please add $10.00 for shipping, handling and insurance. California residents add 6% Sales Tax.
• MAIL ORDER LINE (714) 891-2587 or TELEX 182274.

VAN NUYS	BURBANK	LAWNDALE	ORANGE COUNTY	IRVINE
(213) 786-7411	(213) 848-5521	(213) 370-4842	(714) 891-2584	(714) 891-2589

In Southern California We're Number One!!

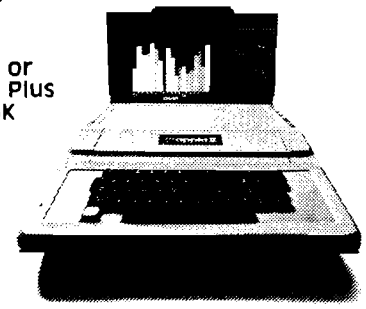
COMPUTER COMPONENTS (CCI) OF ORANGE COUNTY, your full support computer store presents the finest in personal computers

Send for our 36 page fully illustrated catalog $3.00. Save 50¢ by sending the top one inch of this ad with your order.

AN INDUSTRY STANDARD

apple computer

Apple II or Apple II Plus with 16K $1195



SPECIAL

13" Color TV compatible with Apple II only 99.00 with system purchase

APPLE II will change the way you think about computers. That's because it is specifically designed to handle the day to day tasks of education, financial planning, building security, scientific calculation, and entertainment. **APPLE II** is appealing and comfortable (like other appliances that make your life easier); and it brings to personal computing a new level of simplicity through hardware and software sophistication.

INTERFACE CARDS	
Prototyping/Hobby Card	$ 24.00
Parallel Printer Interface Card	180.00
Communications Card & DB25 Connector Cable	225.00
High-Speed Serial Interface Card	195.00
Language System with Pascal (48K RAM & Disk II Required)	495.00
Applesoft II Firmware Card	200.00
16 Input Analog Card	295.00
ACCESSORIES	
Disk II—Drive Only	495.00
Disk II—Drive & Controller (32K Min. RAM Recommended)	595.00
Vinyl Carrying Case	30.00
Tape Recorder	40.00
Programmers Aid No. 1 Firmware (For Use with Integer BASIC)	50.00
Clock/Calendar Card	199.00
Auto-Start ROM Package (For Apple II Only)	65.00
Digikitizer Pad by Talos (Kitform)	499.00
High Resolution Light Pen	199.00
Micromodem II (D.C. Hayes)	379.00
12" B/W Leedex Monitor	149.00
Cable from Monitor to Apple II	9.95
13" Color TV Compatible with Apple II	290.00
Sup-R-Modulator (RF)	25.00
SOFTWARE FOR APPLE II	
PASCAL from Programma	49.95
FORTH	49.95
LISP—from Apple Software 8k No. 3	N/C
LISA—interactive disk assembler	34.95
WHATSIT—Excellent conversational data base manager 32K 100.00 48K 125.00	
SARGON-Champ of 2nd West Coast Computer Faire	19.95
APPLE PIE—Excellent text editor	24.95
FORTE—Music editor in hires	19.95
FASTGAMMON—Excellent backgammon game with graphics Tape 20.00 Disk 25.00	
APPLE 21—Excellent blackjack game	9.95
BRIDGE CHALLENGER—Computer bridge	14.95
FINANCIAL MANAGEMENT SYSTEM	
• Accounts Payable • Ledger Processing	
• Accounts Receivable • Payroll	
• Inventory Control • $800 Complete	
• $200 Each Package • $10 for Manual	

PRINTER SPECIALS FOR APPLE AND PET	
TRENDCOM 100 with interface for Apple or PET	450.00
LITE PEN used with TV or monitor screen	34.95
ALF Music Synthesizer Boards	265.00
Supertalker	279.00
Anadex DP-8000 with tractor 8" paper width and interface to Apple	1050.00
Centronics 779-2 for Apple II with parallel interface	1245.00
SOFTWARE (Send for complete Software Catalog $1.00)	
Dow Jones Portfolio Evaluator/ Stock Quote Reporter Disk	50.00
Microchess 2.0 Chess Disk	25.00
Disk Utility Pack with DOS 3.2	25.00
The Controller (General Business System)	625.00
Apple Post (Mailing List System)	49.95
Bowling Program Diskette	15.00
The Cashier (Retail Store Management	250.00
Checkbook Cassette	20.00
Applesoft II Language & Demo Cassette	20.00
RAM Test Tape with Manual	7.50
Finance 1-2 Cassette Package	25.00
Datamover/Telepong Cassette (Com. Card & Modem Req'd)	7.50
Microchess 2.0 Chess Tape	20.00
Bowling Program Tape	15.00
Pascal with Language System (48K RAM & Disk II Required)	495.00
MISCELLANEOUS	
Vinyl Diskette Holder Pages (Pkg of 10)	8.50
Diskette Boxes (BA Plastic)	4.50
Diskettes (5")	
Apple (Box of 10)	50.00
Verbatim (Box of 10)	34.00
Dysan (Box of 5)	25.00
Decals (Rainbow Apple)	
Inside Window (2")	10¢
Outside Window (2")	10¢
Outside Window (1")	5¢
Apple Logo (Rainbow) T-Shirts Specify—Men, Women, or Child (small, medium, or large)	6.00

Reference Books for Apple and PET Owners

Programming Manual (MOSTEK)	$12.00
Hardware Manual (MOSTEK)	12.00
Programming the 6502 (ZAKS)	9.95
6502 Application Book (ZAKS)	12.95
Programming a Micro Computer: 6502 (FUSTER)	9.95

PET Owners Only

PET User Manual	$9.95
Hands on BASIC with a PET	14.95
PET Machine Language Guide	9.95

Apple Owners Only

Apple II Reference Manual	$10.00
Apple Soft Manual	10.00
Programmer's Guide (Computer Station)	5.95
Apple II Monitor Peeled	9.95
Software Directory for Apple	
• Business, Finance & Utility	4.95
• Games, Demo, Utility	4.95
Best of Contact '78	2.50
Programming in PASCAL (Grogono)	9.90

A PROFESSIONAL BUSINESS SYSTEM

commodore CBM™

2001-16B $995

Also
2001-8 ..... $795
2001-16N ... $995
2001-32N ... $1295
2001-32B ... $1295

341K DUAL DRIVE**

CBM 2040 $1295

Also
External Cassette $95
PET to IEEE Cable $39.95

80 COL. DOT MATRIX PRINTER

CMB 2022 $995

Also
CBM 2023 Printer $849
IEEE to IEEE Cable $49.95

**Retrofit kit required for operation with PET 2001-8.

ACCESSORIES FOR PET

Commodore PET Service Kit	$30.00
Beeper—Tells when tape is Loaded	24.95
Petunia—Play music with PET	29.95
Video Buffer—Attach another display	29.95
Combo—Petunia and Video Buffer	49.95

SOFTWARE FOR PET

Mirrors and Lenses	19.95	Checkers and Baccarat	7.95
The States	14.95	Chess	19.95
Real Estate 1 & 2	59.95	Series and Parallel Circuit Analysis	19.95
Momentum and Energy	19.95	Home Accounting	9.95
Projectile Motion	19.95	BASIC Math	29.95
Mortgage	14.95	Game Playing with BASIC	
Dow Jones	7.95	Vol. I, II, III	9.95 each
Petunia Player Sftwr	14.95		

Plus many more. Send for Software Catalog $1.00.

—Prices subject to change—

COMPUTER COMPONENTS OF ORANGE COUNTY

6791 Westminster Ave., Westminster, CA 92683 714-891-2584 Telex 182274

Hours: Tues-Fri 11:00 AM to 8:00 PM—Sat 10:00 AM to 6:00 PM—Sun 12:00 to 4:00 PM (Closed Mon)

Master Charge, Visa, B of A are accepted. Allow 2 weeks for personal check to clear. Add $2.00 for handling and postage. For computer systems please add $10.00 for shipping, handling and insurance. California residents add 6% Sales Tax.

input of the oscilloscope to PB7 to monitor the square wave. You can use the program to calibrate your oscilloscope sweep time. If you have a frequency counter, measure the frequency of the square wave at PB7 to verify the equation, using the values for N given in Table VI. N is the number to be loaded into T1.

Note that the frequency of the square wave produced at PB7 by the program listed in Table V is as precise as the crystal oscillator frequency used for the system clock. This is because the square wave frequency is independent of any instruction length. The principal advantage of the free-running mode of the T1 timer is that the time between interrupt flag settings (or the frequency of the square wave on PB7) is independent of any instruction length. Thus, one can construct very precise time-keeping routines (*MICRO*, March 1979, pg. 5) or time measuring routines.

To produce simple delay loops for long time intervals, the pulses from PB7 are fed to PB6. Timer T1 operates in its free-running mode, and timer T2 operates in its pulse counting mode. Consequently, T2 counts the pulses produced by T1 on PB7. A program to produce a delay of one hour is given in Table VII. This program may be easily modified to produce delays of 1, 10, 60, 100, 1000, 10000, 36000, or 65536 seconds.

Timer T1 produces a square wave whose period is 0.1 second. These pulses are counted by the T2 counter/timer. If nine is loaded into T2, then 10 pulses, each of 0.1 second duration, will be counted, giving a delay of one second. Other time intervals are programmed accordingly. Of course, there is an uncertainty of several microseconds in the actual loop time, but this uncertainty will be unimportant for most applications.

If the program in Table VII is modified to allow T2 to produce interrupt requests (IRQs) by loading $A0 into the interrupt enable register (IER) at location $A00E (refer to Table IV), then it could be used in connection with the interrupt routine given in Table IV to produce a 24-hour clock program. To generate an interrupt every minute, as required by the low-overhead clock, T1 should count to 600. Load T1 with $0257 instead of $C39F as shown in Table VII and your clock should run. These modifications are shown in the AIM 65 disassembly format.

Sound Effects

The T1 timer can be used in its free-running mode to toggle PB7, and PB7 can be used to drive an amplifier. If the frequency is in the audible range, then a tone will be heard. A series of tones may make up a song. Table VIII lists the frequencies necessary to produce three oc-

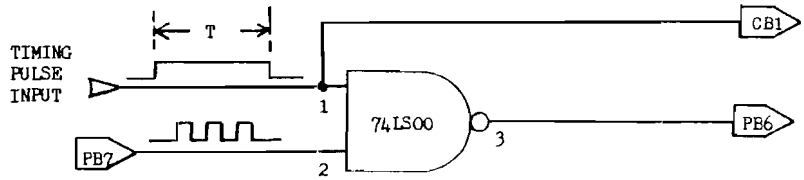


Figure 6. Circuit to measure the time duration, T, of a positive pulse. The CB1 pin must be programmed to produce an interrupt on the negative transition of the pulse by loading PCR4 with a zero. Change the byte at $0217 from $10 to $00 in the listing in Table X to accomplish this.

```
$0200 A9 C0      START  LDA $C0      Set bits seven and six of the ACR,
$0202 8D 0B A0          STA ACR      putting the T1 timer in its free-running
$0205 A9 4D             LDA $4D      mode with a square wave output on PB7.
$0207 8D 06 A0          STA T1LL     Let N = $004D.  Tp = 2($50) microseconds
$020A A9 00             LDA $00      = 160 microseconds.
$020C 8D 05 A0          STA T1LH     Start timer.
$020F 4C 0F 02  LOOP    JMP LOOP     Dummy loop simulates remainder of a program.
```

Table V. Program to Produce a Square Wave Output on PB7.

FREQUENCY	PERIOD	N + 2	N
f	T_p	Decimal Hex	Hex
10 Hz	0.10 sec	50000 = $C350	$C34D
100 Hz	0.01 sec	5000 = $1388	$1386
1000 Hz	1.00 ms	500 = $01F4	$01F2
10 kHz	0.10 ms	50 = $0032	$0030
100 kHz	0.01 ms	5 = $0005	$0003
250 kHz	4.00 us	2 = $0002	$0000

Table VI. Table for Producing Various Square Wave Frequencies.

```
$0200 A9 E0      START  LDA $E0      Load ACR to put T1 in free-running mode
$0202 8D 0B A0          STA ACR      and T2 in pulse counting mode.
$0205 A9 4D             LDA $4D      Initialize T1 timer to run with a period
$0207 8D 06 A0          STA T1LL     of 2($C34D + 2) = 100000 microseconds
$020A A9 C3             LDA $C3      = 0.1 second.
$020C 8D 05 A0          STA T1LH     Start timer toggling PB7.
$020F A9 9F             LDA $9F      Set up T2 to count $C9F + 1 = 36000
$0211 8D 08 A0          STA T2LL     counts. (36000)(0.1sec) = 1 hour.
$0214 A9 8C             LDA $8C
$0216 8D 09 A0          STA T2CH     Start counting.  Clear IFR.
$0219 A9 20             LDA $20      Check interrupt flag register to see if
$021B 2C 0D A0  TEST    BIT IFR      bit five has been set, indicating that
$021E F0 FB             BEQ TEST     T2 has counted 36000 pulses.
$0220 00                BRK          Break to the monitor at the end of an hour
```

Table VII. Program to Produce a One-Hour Delay.

taves of notes on the equally tempered scale (note middle A corresponds to 440 Hz and successive note frequencies are related by a factor equal to the 12th root of two). Also listed in Table VIII are the half periods in microseconds; that is, the numbers that must be loaded into the T1 timer to produce the notes. Since the period of the square wave is $(N + 2)T_C$, each of the numbers in the last column of Table VIII should be decremented by two.

A program to play songs using the notes in Table VIII is listed in Table IX. The identification numbers (I.D. numbers) of the notes in the song to be played are stored in a song table starting at $0400. Actually, the song could be stored anywhere in memory that is convenient, simply by changing the base address of the song table. The base address of the song table is stored in $0050 and $0051, called SONG and SONG + 1, respectively.

Table VIII. Note Table for Producing Tones on the Equally Tempered Scale.

I.D. NUMBER		NOTE	FREQUENCY	PERIOD/2
Hex			Hertz	Microseconds
$00		C_0	130.813	$0EEE
$01		$C_0\#$	138.591	$0E18
$02		D_0	146.832	$0D4D
$03		$D_0\#$	155.563	$0C8E
$04		E_0	164.814	$0BDA
$05		F_0	174.614	$0B2F
$06		$F_0\#$	184.997	$0A8F
$07		G_0	195.998	$09F7
$08		$G_0\#$	207.652	$0968
$09		A_0	220.000	$08E1
$0A		$A_0\#$	233.082	$0861
$0B		B_0	246.945	$07E9
$0C	(middle)	C_1	261.626	$0777
$0D		$C_1\#$	277.183	$070C
$0E		D_1	293.665	$06A7
$0F		$D_1\#$	311.127	$0647
$10		E_1	329.628	$05ED
$11		F_1	349.228	$0598
$12		$F_1\#$	369.995	$0548
$13		G_1	391.995	$04FC
$14		$G_1\#$	415.304	$04B4
$15		A_1	440.000	$0470
$16		$A_1\#$	466.164	$0431
$17		B_1	493.883	$03F4
$18		C_2	523.251	$03BC
$19		$C_2\#$	554.365	$0386
$1A		D_2	587.330	$0353
$1B		$D_2\#$	622.254	$0323
$1C		E_2	659.255	$02F6
$1D		F_2	698.456	$02CC
$1E		$F_2\#$	739.989	$02A4
$1F		G_2	783.991	$027E
$20		$G_2\#$	830.609	$025A
$21		A_2	880.000	$0238
$22		$A_2\#$	932.328	$0218
$23		B_2	987.767	$01FA

The identification numbers ($00 - $23) found in the song table are used to index a note table found in page zero, from $0000 to $0047. The note table contains the half-periods of the frequencies found in the fourth column of Table VIII, corrected for the fact that the half-period is $(N + 2)T_C$ rather than $(N)T_C$. The low-order bytes of the half-periods are found from $0000 to $0023 in the note table, while the high-order bytes are found from $0024 to $0047.

The program first locates an identification number for a note from the song table. It then loads the latches on the T1 timer with the correct half period, and the note begins to play. The duration of the note is determined by a number found in the duration table, called DUR, and located from $0800 upward. There must be one duration number for each note. The duration of a note is basically the number of times the T2 timer is allowed to time out. If $01 represents a sixteenth note, then $02 is an eighth note, $04 is a quarter note, $08 a half note, and $10 a whole note. The tempo may be changed by changing the bytes loaded into the T2 timer at locations $021E through $0227 in the program listed in Table IX.

The song table given in Table IX simply plays the three octave scale from Table VIII with a variety of durations as indicated by the duration table. You are invited to make your own song or translate someone else's song into I.D. numbers. Better yet, write a song interpreter that does the translation for you.

Your interpreter should take a keyboard entry for a note and place the I.D. number into the song table. It should take another keyboard entry for the time value of the note and place it in the duration table. With several 6522s, you could play four-part harmony! With a D/A converter and a voltage controlled amplifier you could also control the note envelopes, giving an elementary synthesizer.

For my interface circuit, I used the 7404 inverter connected to PB7. The output from the 7404 was connected to one lead of a 1½ inch speaker and the other lead was connected to +5 volts. Better interfacing circuits to drive speakers have appeared in various articles and books (see Caxton Foster's *Programming a Microcomputer*).

Measuring the Time Between Events

A number of applications require that the time between two successive events be measured. The events might be the start and finish of a race, the arrival of cosmic rays, two heartbeats of an animal, and many others. If the events are periodic, then the time between events can be obtained by first measuring the frequency of the events with a

Table IX. Program to Play a Song.

```
$0050 = SONG, [SONG] = $00
$0051 = SONG + 1, [SONG + 1] = $04
$0052 = DUR, [DUR] = $00
$0053 = DUR + 1, [DUR + 1] = $08
$0000 = NOTE (See Note Table)

NOTE TABLE
$0000 EC 16 4B 8C D8 2D 8D F5
$0008 66 DF 5F E7 75 0A A5 45
$0010 EB 96 46 FA B2 6E 2F F2
$0018 BA 84 51 21 F4 CA A2 7C
$0020 58 36 16 F8 0E 0E 0D 0C
$0028 0B 0B 0A 09 09 08 08 07
$0030 07 07 06 06 05 05 05 04
$0038 04 04 04 03 03 03 03 03
$0040 02 02 02 02 02 02 02 01

DURATION TABLE
$0800 01 02 04 08 10 20 10 08
$0808 04 02 01 02 04 08 10 20
$0810 10 08 04 02 01 02 04 08
$0818 10 20 40 80 40 20 10 08
$0820 04 02 01 01 00

SONG TABLE (Plays scale)
$0400  00 01 02 03 04 05 06 07
$0408  08 09 0A 0B 0C 0D 0E 0F
$0410  10 11 12 13 14 15 16 17
$0418  18 19 1A 1B 1C 1D 1E 1F
$0420  20 21 22 23
```

```
$0200 A9 C0     START   LDA $C0      Initialize ACR to put T1 in free-running
$0202 8D 0B A0          STA ACR      mode.
$0205 A0 00             LDY $00      Indirect indexed mode with index = 0.
$0207 B1 50     MORE    LDA (SONG),Y Get note I.D. from song table.
$0209 AA                TAX          Use it as an index to look up note
$020A B5 00             LDA NOTE,X   in the note table.
$020C 8D 06 A0          STA T1LL     Put low-order byte into T1LL
$020F 8A                TXA          Transfer X back to A to find high-order
$0210 18                CLC          byte, which is $24 locations higher
$0211 69 24             ADC $24      in page zero.
$0213 AA                TAX          Back into X to become index to fetch
$0214 B5 00             LDA NOTE,X   high-order byte of half-period.
$0216 8D 05 A0          STA T1LH     Result into T1 timer latch high.  Note
$0219 B1 52             LDA (DUR),Y  begins to play.  Get duration.
$021B F0 24             BEQ OUT      If duration is zero, end of song.
$021D AA                TAX          Duration into X to serve as counter.
$021E A9 FF     AGN     LDA $FF      Set up T2 for a time period that determines
$0220 8D 08 A0          STA T2LL     the tempo.
$0223 A9 FF             LDA $FF
$0225 8D 09 A0          STA T2CH     Start the T2 timer.
$0228 A9 20             LDA $20      Test to see if T2 has timed-out.
$022A 2C 0D A0  BACK    BIT IFR      Is bit five of the IFR set?
$022D F0 FB             BEQ BACK     No, wait for it and play note.
$022F CA                DEX          Decrement duration counter until
$0230 D0 EC             BNE AGN      it is zero, then note is finished.
$0232 E6 50             INC SONG     Get another note from the song table.
$0234 D0 02             BNE PAST     If song is zero, then get the next note from
$0236 E6 51             INC SONG + 1 next page of song table.
$0238 E6 52     PAST    INC DUR      Get another duration from the table.
$023A D0 02             BNE THERE
$023C E6 53             INC DUR + 1
$023E 4C 07 02  THERE   JMP MORE     Play this note.
$0241 A9 00     OUT     LDA $00      Clear the ACR to finish playing notes.
$0243 8D 0B A0          STA ACR
$0246 00                BRK          Jump to the monitor when finished.
```

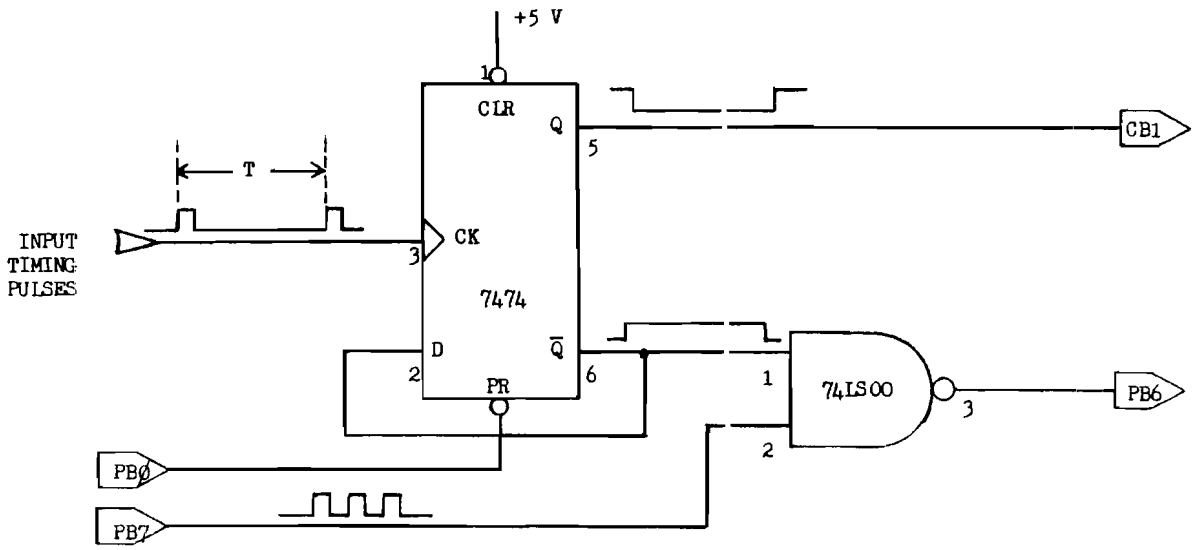


Figure 4. Circuit to measure the time interval, T, between two successive pulses.

Table X. Program to Measure the Time Between Two Pulses.

```
$0200 A9 00     START   LDA $00       Clear display registers.
$0202 85 01             STA LEAST     Least-significant  byte of time.
$0204 85 02             STA MIDST     Middle byte.
$0206 85 03             STA MOST      Most-significant byte of time.
$0208 A9 01             LDA $01       Initialize PBØ to be an output pin.
$020A 8D 02 A0          STA DDRB
$020D 8D 00 A0          STA PBD       Initialize PBØ to logic one, then toggle
$0210 CE 00 A0          DEC PBD       it to preset the 7474 flip-flop.
$0213 EE 00 A0          INC PBD
$0216 A9 10             LDA $10       Set bit four of the peripheral control
$0218 8D 0C A0          STA PCR       register (PCR) to set interrupt flag on
$021B A9 E0             LDA $E0       a positive transition on pin CB1.
$021D 8D 0B A0          STA ACR       T1 in free-running mode, T2 counts pulses.
$0220 A9 86             LDA $86       Set period of square wave on PB7 so that
$0222 8D 06 A0          STA T1LL      Tp = 0.01 second.
$0225 A9 13             LDA $13       $1386 + 2 = 5000, so f = 100 Hz, Tp = 0.01s.
$0227 8D 05 A0          STA T1LH      Start square wave running.
$022A A9 FF     NEXT    LDA $FF       Set up pulse counter T2 to start at $FFFF.
$022C 8D 08 A0          STA T2LL
$022F 8D 09 A0          STA T2CH      Start counting pulses when the event pulse
$0232 AD 00 A0          LDA PBD       clocks the 7474 flip-flop.  Clear IFR4 flag.
$0235 AD 0D A0  TEST    LDA IFR       Read the interrupt flag register.  Mask
$0238 29 10             AND $10       all except IFR4.  Wait until flag is set,
$023A F0 F9             BEQ TEST      then timing is finished, so convert the
$023C 20 00 03          JSR CNVD      answer to decimal and display it.
$023F CE 00 A0          DEC PBD       Preset the flip-flop by toggling PBØ.
$0242 EE 00 A0          INC PBD
$0245 4C 2A 02          JMP NEXT      Measure another interval.

SUBROUTINE CNVD
$0300 38        CNVD    SEC           Set carry for subtractions that follow.
$0301 A9 FF             LDA $FF       Find ($FFFF - N2) = number of pulses counted.
$0303 ED 09 A0          SBC T2CH
$0306 85 11             STA CNTHI     High-order byte stored in CNTHI.
$0308 A9 FF             LDA $FF       Now get the low-order byte of the count.
$030A ED 08 A0          SBC T2CL
$030D 85 10             STA CNTLO     Low-order byte stored in CNTLO.
$030F F8                SED           Conversion of hex to decimal starts here.
$0310 A0 10             LDY $10       Y contains number of bits to convert.
$0312 06 10     MORE    ASL CNTLO     Shift one bit at a time into the carry flag.
$0314 26 11             ROL CNTHI
$0316 A2 FD             LDX $FD       X will serve as a counter for a triple-
$0318 B5 04     AGIN    LDA DAT,X     precision addition, with LEAST, MIDST,
$031A 75 04             ADC DAT,X     and MOST holding the answer.
$031C 95 04             STA DAT,X
$031E E8                INX           Increment X to zero, then three bytes
$031F D0 F7             BNE AGIN      have been added.
$0321 88                DEY           Decrement Y until all the bits have been used.
$0322 D0 EE             BNE MORE      When Y = 0, conversion is complete.
$0324 20 40 03          JSR AIMDSP    Jump to AIM 65 Display Routine.
$0327 A9 00             LDA $00       Now clear the counter locations to get
$0329 85 01             STA LEAST     the time for the next two pulses.
$032B 85 02             STA MIDST
$032D 85 03             STA MOST
$032F 60                RTS           Return to the timing program.
```

frequency counter and then applying the relation T = 1/f, where T is the time between successive events and f is the frequency of the events. For low frequency periodic events, such as a race, the only choice is to measure the time interval directly.

We will assume that the events produce positive pulses, and we will not try to describe how the positive pulses can be produced. Rather, our problem will be restricted to measuring the time between two successive positive pulses. A circuit and a program to accomplish this are shown in Figure 4 and Table X, respectively.

The circuit was inspired by Carlin's and Howard's article on the Intel 8253 in *Computer Design*, May 1979, pg. 213. The positive pulses clock a 7474 flip-flop, producing a logic-one voltage at the Q output of the 7474 for the time interval between the leading edges of the two pulses. With the T1 timer producing square waves on PB7, the logic-one voltage on the Q output gates the pulses to PB6 (by means of the 7400 NAND gate), where they are counted by the T2 counter/timer. For example, if a square wave whose frequency is 10 Hz (T = 0.1 second) is applied to the 7400 NAND gate, and 250 such pulses are counted on PB6, then the corresponding time interval is (250)(0.1) = 25.0 seconds, with a resolution of 0.1 second.

Clearly, no software is required to detect the pulses, and consequently very narrow pulses can be detected. Also, the programmer has control over the frequency of the square wave applied to the NAND gate. The resolution can be changed from 4.0 microseconds to 0.10 microseconds by varying the number loaded into T1.

Refer again to Table VI for a choice of frequencies for the free-running mode of the T1 timer that might be appropriate for a given application. Since the T2 timer is capable of counting to 65536, the maximum time interval that can be measured with a square wave whose period is T_p is:

$$T_{max} = 65536(Tp)$$
$$= 65536(2)(N + 2)T_c$$

where T_{max} is the maximum time interval that can be measured, T_p is the period of the square wave (T_p = 1/f) on PB7, N is the number loaded into T1, and T_c is the system clock period.

Refer again to Figure 4. When the second pulse occurs, the Q output of the 7474 flip-flop makes a transition to logic one. This also signals the conclusion of the timing interval. If Q is connected to CB1, the 6522 can be programmed to set a flag in the IFR when the logic-zero-to-logic-one transition on CB1 occurs. At this time the T2 counter/timer can be read, the result converted to decimal,

APPENDIX A. LOW—OVERHEAD CLOCK MODIFICATION

```
SUBROUTINE AIMDSP

$0340 A5 LDA 01
$0342 85 STA 04
$0344 A5 LDA 02
$0346 85 STA 05
$0348 A5 LDA 03
$034A 85 STA 06
$034C A2 LDX #13
$034E 8A TXA
$034F 48 PHA
$0350 A0 LDY #04
$0352 A5 LDA 04
$0354 29 AND #0F
$0356 18 CLC
$0357 69 ADC #30
$0359 09 ORA #80
$035B 20 JSR EF7B
$035E 46 LSR 06
$0360 66 ROR 05
$0362 66 ROR 04
$0364 88 DEY
$0365 D0 BNE 035E
$0367 68 PLA
$0368 AA TAX
$0369 CA DEX
$036A E0 CPX #0E
$036C B0 BCS 034E
$036E 60 RTS
```

and the answer can be displayed or logged for the next set of pulses. All of this is accomplished with the routines given in Table X, a program that was designed to operate in conjunction with the circuit of Figure 4. An explanation of this program follows.

The largest number of pulses from PB7 that can be counted on pin PB6 by the T2 counter/timer is $FFFF + 1 or 65536. Each memory location is capable of storing two BCD digits, thus three memory locations are required to store a number as large as 65536. These three memory locations have addresses $0001 through $0003 in the program shown in Table X, and they are used to store the decimal equivalent of the count made by the T2 counter/timer. The initialization steps, display registers cleared, flip-flop preset, timers loaded, control registers set, etc., require the first $34 bytes in the program. After that, the interrupt flag register (IFR) is watched to see when a positive transition on CB1 occurs. When it does, a jump to the conversion subroutine, CNVD, occurs.

The function of the conversion subroutine is to convert the contents of the T2 counter/timer registers to an actual count in decimal. This count represents the number of periods of the square wave on PB7 that have occurred between the events being timed. The program in Table X uses a square wave whose period is 0.01 seconds, thus the number of counts in T2 represents the number of hundredths of seconds that occurred between the two positive pulses on the clock input of the 7474 flip-flop.

```
0200 78 SEI
0201 A9 LDA #A0
0203 8D STA A00E
0206 A9 LDA #E0
0208 8D STA A00B
020B A9 LDA #4D
020D 8D STA A006
0210 A9 LDA #C3
0212 8D STA A005
0215 A9 LDA #57
0217 8D STA A008
021A A9 LDA #02
021C 8D STA A009
021F 58 CLI
0220 4C JMP 0220

INTERRUPT
ROUTINE  0300 A9 LDA #02
         0302 8D STA A009
         0305 18 CLC
         0306 F8 SED
         0307 A5 LDA 00
         0309 69 ADC #01
         030B 85 STA 00
         030D C9 CMP #60
         030F D0 BNE 0324
         0311 A9 LDA #00
         0313 85 STA 00
         0315 18 CLC
         0316 A5 LDA 01
         0318 69 ADC #01
         031A 85 STA 01
         031C C9 CMP #24
         031E D0 BNE 0324
         0320 A9 LDA #00
         0322 85 STA 01
         0324 D8 CLD
         0325 40 RTI
```

The time between the leading edges of the positive pulses produced by the events (call this time T) as *measured* by the program in Table X is given by the formula:

$$T_m = T_p(\$FFFF - N_2)$$
$$= 2(N_1 + 2)(\$FFFF - N_2)t_c$$

where T_p is the period of the square wave on PB7, N_2 is the number in the T2 counter/timer at the conclusion of the timing interval, and N_1 is the number in the T1 timer. Refer to Table VI for the necessary N_1 to produce a suitable T_p. Values of T_p that are multiples of ten are most useful. The origin of the number $FFFF in the equation lies in the fact that the T2 counter/timer is loaded with $FFFF before timing begins. For the listing shown in Table X, T_p is 0.01 seconds, so the equation becomes:

$$T_m = 0.01(\$FFFF - N_2) \text{ seconds}$$

The precision with which one can measure the true time T between the events depends on the resolution, T_p, since clearly the true time need not be an exact integral number of T_p. Our analysis shows that the actual time, T, is given by the expression:

$$T_m - 1\frac{1}{2}T_p \leq T \leq T_m + \frac{1}{2}T_p$$

Thus, if greater precision is required, then T_p can be reduced.

The conversion subroutine, CNVD, performs the operation ($FFFF - N_2) shown in the equations. To get T, this number must be converted to decimal and then multiplied by T_p which, in our case, is 0.01 seconds. The hexadecimal to decimal conversion algorithm used in CNVD is from Peatman's book *Microcomputer Based Design*, while the coding used is from Butterfield's "Multi-Mode Adder" in *6502 User Notes*, No. 13, pg. 23.

Subroutine CNVD also calls a subroutine named AIMDSP. This routine displays the contents of locations with addresses $0001, $0002, and $0003; namely those locations that contain the time T, now in decimal. No attempt has been made to locate the decimal point in these subroutines. As long as the period, T_p, if the square wave on PB7 is a multiple of ten, 0.01 second for example, the user should have no trouble placing his decimal point mentally.

In any case, subroutine AIMDSP is an AIM 65 dependent subroutine that has been published previously, so only its AIM 65 mini-disassembly format is given here. Owners of other microcomputer systems will want to substitute a suitable routine to display the contents

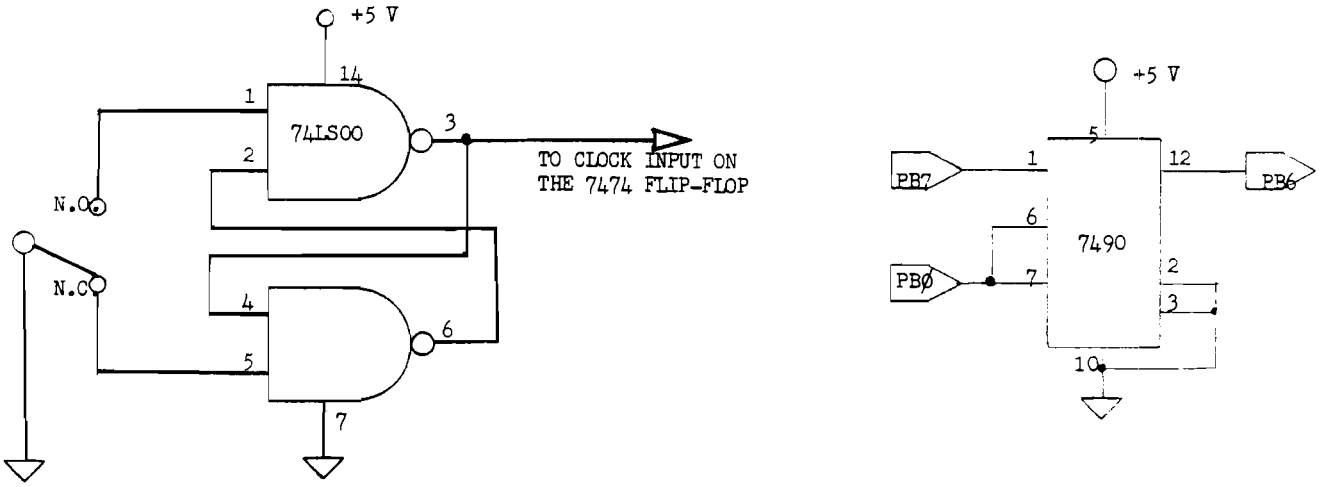


Figure 5. Stopwatch Interface for the Circuit in Figure 4. The switch is normally closed (N.C.). To produce a pulse when an event occurs, the normally open (N.O.) contact is closed momentarily.

of the three locations mentioned. Such routines for the KIM-1 and SYM-1 are readily available.

The time interval chosen for the listing in Table X is suitable for "stopwatch" functions, and a suitable stopwatch interface to the circuit of Figure 4 is given in Figure 5. This circuit simply debounces the switch when it is *momentarily* closed at the beginning and the end of the interval to be timed. Phototransistor circuits can also be used to produce positive pulses when light beams are interrupted. A photoplethysmograph can be used to measure the time interval between heartbeats, turning the circuit of Figure 4 into a cardiotachometer.

One way to test the circuit of Figure 4 and the program in Table X is to apply a square wave of known frequency to the clock input on the 7474. For example, if the pulses from the signal conditioner shown in Figure 3 are applied to the 7474, then the time interval should be 1/60 of a second. Since 1/60 = 0.01666, and if T_p = 0.0001 second (N_1 = $0030 from Table VI), then the number 1666 should be displayed for the time between successive positive pulses. Be sure to change the bytes at $0221 and $0226 to $30 and $00, respectively, in Table X if you make this test.

Finally, if an event can be made to produce a single positive pulse for its duration, the length of the event may be measured using a slightly modified form of the program in Table X and the circuit shown in Figure 6.

In conclusion I should like to point out that the programs and circuits given are the simplest ones I could construct. You will want to add more elegant features. The purpose of this article was to introduce a few basic techniques, not to present elaborate designs. If you come up with a neat design as a result of something you learned here, I would be very interested in getting a letter from you. Better yet, write up your circuit and program and publish both in *MICRO*. Although the circuits and programs described here were intended to be building blocks for more elaborate microprocessor based designs, the stopwatch interface and timing program could be used for "time and motion" studies around the house. Just make sure your spouse's motions do not make you lose track of the time!

Editor: Portions of this article are from Dr. De Jong's forthcoming book tentatively entitled 6502 Microcomputing, *to be published by Howard W. Sams and Company, and scheduled for release later this autumn.*

```
0200 A9 LDA #01     Set up the Port B DDR with a one in bit zero.
0202 8D STA A002
0205 8D STA A000    Start with pin PBØ = 1 to preset 7490.
0208 CE DEC A000    Allow 7490 to count.
020B A9 LDA #E0     Initialize ACR to put T1 in free-running mode, T2 counts
020D 8D STA A00B
0210 A9 LDA #4D
0212 8D STA A006    Frequency of square wave on PB7 = 10 Hz, Tp = 0.1 second.
0215 A9 LDA #C3
0217 8D STA A005    Start T1 running.
021A A9 LDA #9F     Set up T2.
021C 8D STA A008
021F A9 LDA #8C     T = 20(N1 + 2)(N2 + 1)Tc
0221 8D STA A009    Start counting.
0224 A9 LDA #A0     Set up interrupt enable register (IER) to allow an
0226 8D STA A00E    interrupt request (IRQ) when T2 times out.
0229 58 CLI
```

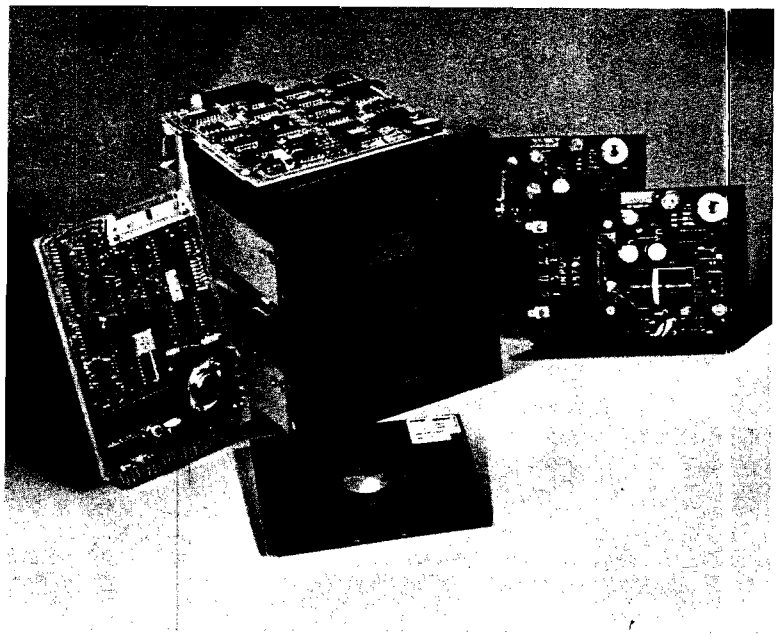
(Note: The interrupt routine should reload T2CH with $8C to clear the IFR and allow counting to proceed again, if equally spaced, 10-hour interrupts are desired.)

HDE inc.

BOX 120
ALLAMUCHY, N.J. 07820
201-362-5574

HUDSON DIGITAL ELECTRONICS INC.

THE HDE MINI-DISK SYSTEM



VERSIONS

KIM
TIM
AIM 65 – 4th Qtr. '79
SYM – 1st Qtr. '80

SINGLE DRIVE	**$ 795.00**
DUAL DRIVE	**$1195.00**

Complete with all hardware. Interconnecting cables, FODS, text editor and user and installation manuals.

The HDE DM816-MD1 Mini Disk System is the peripheral you have been waiting for. No longer bounded by long and unreliable cassette saves and loads, your computer becomes a sophisticated system for program development or general purpose use. With the HDE Mini-Disk you load and save programs in seconds, not minutes or hours. And, since all transfers to and from the Mini-Disk are verified for accuracy, the data will be there when you need it.

The HDE DM816-MD1 Mini-Disk has been "systems" engineered to provide a complete and integrated capability. Software and hardware have been built as a team using the most reliable components available. The systems software includes the acclaimed and proven HDE File Oriented Disk System and Text EDitor, requiring only 8K for the operating software and overlay area. Systems expanding programs available include the two-pass HDE assembler, the Text Output Processing System and Dynamic Debugging Tool. Hardware includes a Western Digital 1771 based controller in a state-of-the-art 4½ x 6½" card size, Shugart SA 400 drive and the Alpha power supply.

The storage media for the DM816-MD1 is the standard, soft sectored 5¼" mini diskette readily available at most computer stores, and HDE has designed the system so that the diskettes rotate only during disk transactions, favorably extending media life. A disk formatter routine included with the system, formats the diskettes, verifies media integrity by a comprehensive R/W test and checks drive RPM. Additional utilities provide ascending or descending alpha numeric sort, disk packing, text output formatting, file renaming, file addressing and other capabilities.

HDE PRODUCTS. BUILT TO BE USED WITH CONFIDENCE.

AVAILABLE DIRECT OR FROM THESE FINE DEALERS:

JOHNSON COMPUTER	PLAINSMAN MICROSYSTEMS	ARESCO	LONG ISLAND COMPUTER GENERAL STORE	LONE STAR ELECTRONICS
Box 523	Box 1712	P.O. Box 43	103 Atlantic Avenue	Box 488
Medina, Ohio 44256	Auburn, Ala. 36830	Audubon, Pa. 19407	Lynbrook, N.Y. 11563	Manchaca, Texas 78652
216-725-4560	800-633-8724	215-631-9052	516-887-1500	612-282-3570

Card Shuffling Program for KIM - 1

Your 6502 might play poker like Amarillo KIM, but does it always have to pass the deal? Not if you teach it to shuffle cards!

Hark Chan
P.O. Box 714
Cambridge, MA 02139

Entertaining friends with computer games certainly makes all the effort of assembling a personal computer worthwhile. However, if you happen to have a small microcomputer with limited memory and very few software tools, there are not many games available. As an example, most card games need a random number generator to shuffle cards.

The standard method to generate random numbers (as used in most BASIC interpreters) is not suitable for this purpose. Since some of the bare-bone computers do not even have the software to perform multiplication, it is asking too much for them to generate floating-point random numbers. To make these small computers more entertaining, a simple method to shuffle cards is described here. This method is implemented in a KIM. The machine instructions use about 80 bytes. There is lots of memory left for playing card games. The only drawback is that it requires the operator to press the interrupt key in order to stop the program.

The card shuffling program consists of two portions. The second portion is the main program that shuffles cards. It just keeps on shuffling until the interrupt key is pressed. The first portion is an interrupt service routine used to ensure an orderly ending of the program. The program is relocatable, and the two portions can be in separate locations.

This feature makes it easy to incorporate the shuffling program into a complete card-playing program. However, it is important that the user initialize the interrupt vectors to jump to the interrupt service routine.

To keep the computer code relocatable, the initialization of the 2 byte address is left to the user. The storage area for the cards, together with 4 bytes of working space, are in page 0. In this program, the storage area starts at address 0001. However, the program can be changed easily to move the storage area to other locations in page 0.

The deck of cards is stored in an array at locations (hex) 0001 to 0034. The value of

```
0120: 0200                ORG   $0200
0130: 0200 A2 36          LDXIM $36
0140: 0202 8A       L1    TXA
0150: 0203 95 00          STAZX $00
0160: 0205 CA             DEX
0170: 0206 D0 FA          BNE   L1
0180: 0208 86 38          STXZ  $38
0190: 020A A5 35    LOOP  LDAZ  $35
0200: 020C 38       L2    SEC
0210: 020D E9 34          SBCIM $34
0220: 020F B0 FB          BCS   L2
0230: 0211 18             CLC
0240: 0212 69 35          ADCIM $35
0250: 0214 AA             TAX
0260: 0215 85 35          STAZ  $35
0270: 0217 B5 00          LDAZX $00
0280: 0219 85 37          STAZ  $37
0290: 021B A5 36          LDAZ  $36
0300: 021D 0A             ASLA
0310: 021E 0A             ASLA
0320: 021F 18             CLC
0330: 0220 65 36          ADCZ  $36
0340: 0222 18             CLC
0350: 0223 69 01          ADCIM $01
0360: 0225 85 36          STAZ  $36
0370: 0227 18             CLC
0380: 0228 65 35          ADCZ  $35
0390: 022A 38       L3    SEC
0400: 022B E9 33          SBCIM $33
0410: 022D B0 FB          BCS   L3
0420: 022F 18             CLC
0430: 0230 69 34          ADCIM $34
0440: 0232 AA             TAX
0450: 0233 B4 00          LDYZX $00
0460: 0235 A5 37          LDAZ  $37
0470: 0237 95 00          STAZX $00
0480: 0239 A6 35          LDXZ  $35
0490: 023B 94 00          STYZX $00
0500: 023D A5 38          LDAZ  $38
0510: 023F C9 00          CMPIM $00
0520: 0241 F0 C7          BEQ   LOOP
```

```
0010:
0020:                    * INTERRUPT SERVICE ROUTINE
0030:
0040: 0243 A5 F3          LDAZ  $F3
0050: 0245 A4 F4          LDYZ  $F4
0060: 0247 A6 F5          LDXZ  $F5
0070: 0249 E6 38          INCZ  $38
0080: 024B 40             RTI
```

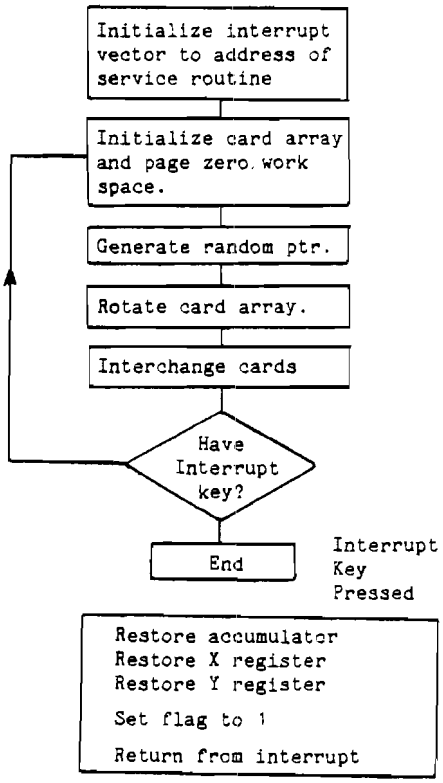
each address is distinct and is between hex 1 to 34 (decimal 1 to 52). After the interrupt key is pressed, the content of these addresses represents a deck of random cards.

The program uses a simple random number generator to generate random pointers with values between 1 and 52. The first card in the deck is interchanged with the card selected by the random pointer. The position of all the cards is next shifted one place so that the last card becomes the first, the first card becomes the second, and so on. This is to make sure that the first card is always changing, and a different card is interchanged with each randomly selected card. A random pointer is again generated and the whole operation is repeated.

After a sufficient number of operations, the deck is suitable for card games. One or two hundred shufflings are sufficient.

When the interrupt key is pressed, the interrupt service routine sets a memory location, hex 0038, that serves as a flag to signal the end of the shuffling. This routine also restores the accumulator and the X and Y registers. It is important that the user initialize the interrupt vector to address the service routine instead of the operating system.

The sequence of cards being shuffled is actually predetermined because it is calculated from a prescribed series of operations. However, if the stop command is activated by a human operator the cards can be very random. It takes about 10^{-4} second to do one shuffle. The time to activate the stop command can easily vary by more than 0.1 second. Thus, the number of shufflings can be uncertain by about 1000, which is sufficient to generate a deck of random cards.



How Do You Connect Peripherals to Your Superboard II

The OSI Superboard has a wealth of I/O ports, but often the effective use of them is "Left as an exercise for the reader". Here is some concise information on the configuration and use of the I/O ports.

Bruce Hoyt
Route 1
Brighton, TN 38011

Since I wrote "A Close Look at the Superboard II", *MICRO* 11:15, I have received several calls and letters asking for more information concerning interfacing the Superboard II to various peripherals — printers, memory boards and so on. Because of the continuing lack of information available from OSI, the manufacturer of the Superboard and the Challenger 1P, I have decided that it would be good to give some basic and rather general pointers on the use of the Superboard ports.

Since there are many different peripherals (understatement of the century) and since each one has its own requirements, I cannot be very specific about your particular device. Instead, I hope to describe the signals available on the Superboard in some detail, so that you will at least know something about its interfacing possibilities.

The J2 Port

There are four ports on the Superboard. Three of them are 12-pin Molex connectors and one of them is a 40-pin DIP socket. They are numbered J1 through J4. I shall begin with J2, since you are already using that one to interface your video monitor and your cassette. You will find a listing of the pin outs for J2 in Figure 1. Pins 7 through 10 are used for the cassette. Pins 11 and 12 are used for the video output.

I assume that you understand the basic use of these pins; and so, I will only mention that the signals generated for the cassette come from an on-board interface consisting of a Motorola 6850 ACIA and a couple of flip flops (U64). The audio input goes through an RCA 3130 which triggers a monostable one-shot and sets or resets a flip flop. This signal is then fed to the 6850.

The signals at the 6850 are designated as RxData and TxData. The 6850 also has two control signals which are not used by the cassette interface but might be useful to your peripheral. They are designated as RTS and CTS on the schematics.

Finally, there are two separate clocks which drive the 6850: TxCLK and RxCLK. These clocks set the baud rate at which the 6850 operates. For precise information on the 6850, I suggest that you get a copy of the manufacturer's spec sheet on this ACIA. Your dealer should have it. I mention all of this simply because these six signals are present as TTL signals on J2, pins 1 through 6. If your peripheral requires TTL level serial data, then you will connect it to these pins.

But there is more to it than just connecting your peripheral's cable to the right pins on J2. My Superboard II came with several parts missing. You will need to install a 7417 at U68 and a 74LS14 at U67. You will also have to instal the 220 and 390 ohm resistors at R38 through R49.

Next, notice that the RxData and CTS signals coming in on pins 1 and 3 respectively are called RxData3 and CTS3 after they come from U67. They are then routed to jumper locations W10 (the upper W10 to the right of Q2 in the schematic sheet 6) and W11. The reason for this is that you don't want input coming from two or three different sources going to the 6850.

I recommend that you install a DP3T (double pole three throw) switch so that you can switch the RxData line going to the 6850 between RxData1, which is the cassette input; RxData3, which is the TTL level input from J2; and the RS-232 input which will be described shortly. The other pole of this switch can be used to switch CTS appropriately. To install this switch you only have to cut the trace connecting the RxData line to RxData1 at W10.

With this switch installed, you can switch lines between three sources of input: the cassette, your peripheral on TTL level lines at J2, and some other peripheral that uses RS-232 on J3.

One more change may be needed at jumper location W5, also on sheet six of the schematic. Here, the TxCLK is wired to the RxCLK. To separate them, you merely have to cut the diagonal trace connecting them and install another switch to switch the RxCLK line on the 6850 between the TxCLK line and the RxCLK input. I recommend, however, that you not make this modification unless you need separate clocks for your peripheral. If your peripheral is pretty stable and close to 300 baud, you can probably get by as is. But if you have a peripheral that has a clock rate different from 300 baud, you will need to make this modification.

You may now ask what the RTS and CTS signals are used for. If your peripheral is a printer, it may send out a busy signal whenever it is not ready to receive another character. This signal should be active high. It should be connected to the CTS on the 6850 — that is, it should be connected to J2 pin 3. You will have to switch W11 properly, since the CTS goes through this junction. You may also have a TTL line which controls the power on/off on your peripheral. Maybe you would like to control the cassette motor. You can do this with the RTS signal. It is a signal provided by the 6850 under software control; that is, *your* software, since OSI doesn't support this function.

Because it is fed through a 7417 buffer which is capable of sinking 30 milliamps, you can use it to drive a small reed relay. I purchased just such a relay, which operates on 5 volts at about 20 milliamps, and have used it to turn my cassette on and off. See Figure 2 for a schematic used to connect a relay to the RTS signal.

Now all the connections are made, but how do you instruct the computer to

transmit and receive these signals? Remember that the cassette is also connected to the 6850; and so, as far as software is concerned, the peripheral will work just like the cassette. Whatever you write to your cassette will go to the TxData line and to your peripheral. You read your peripheral just as you would read from the cassette (after you switch W10 over).

Let us suppose that you have a printer connected to the TxData line and that it sends a busy signal back over the CTS line when it is working. Whenever you give the command to "SAVE" in BASIC, this will activate the printer just as it does the cassette, so that any characters output by BASIC will be sent to both printer and cassette. If either of them is turned on, it will print or record the data sent. And how can one tell whether the printer is busy or not? You can't without writing some of your own software.

You see, Microsoft BASIC does not actually do any I/O; it merely jumps out to the I/O routine provided by OSI in the monitor. There are four routines that BASIC jumps to for I/O: one which inputs a character, one which outputs a character, one which is executed whenever the LOAD command is given, and one which is executed whenever the SAVE command is given. BASIC jumps to the following addresses which have instructions as shown:

Input	FFEB	JMPI	$0218
Output	FFEE	JMPI	$021A
Load	FFF4	JMPI	$021E
Save	FFF7	JMPI	$0220

The monitor stores the addresses of the input, output, load, and save routines at the locations $0218, $021A, $021E, and $0220 respectively every time the BREAK key is pressed. This makes BASIC transfer control to these routines when it needs I/O.

Of course, it would be easy to write your own routine and POKE the address of it in one of these locations so BASIC would then jump to your routine instead of the one in the monitor. You can disassemble the routines in the monitor, if you want to find out just what they do, but I will describe their functions here.

The input routine, located at $FFBA, checks the load flag at $0203. If it is zero, the routine jumps to the keyboard input routine at $FD00 to input a character from the keyboard. If the flag is non-zero, the input routine checks to see if the spacebar is pressed and, if not, it inputs one character from the 6850 and returns. If the spacebar is pressed, it sets the load flag to zero and inputs a character (which will be a space since the spacebar is pressed) from the keyboard. This is why pressing the spacebar will stop reading from the cassette.

The output routine, located at $FF69, jumps to the CRT simulator routine at $BF2D which outputs a character to the screen and then checks the save flag at $0205. If the save flag is 0 it returns. If the save flag is non-zero, it outputs the character to the 6850. If this character was a carriage return (that is, $0D) then it also sends out 10 nulls ($00).

The load routine, located at $FF96, sets the save flag to 1. When you give the SAVE command, BASIC jumps to the save routine which sets the save flag. Then, whenever you output any character, BASIC jumps to the output routine which sends the character not only to the CRT, but also to the 6850. This will send it to the cassette and also to your printer. If you don't turn on your cassette, the character will only be printed by the printer.

But I still haven't described how you know when the printer is busy. You can PEEK at the 6850 control status register to see whether the CTS bit is low. Then you will know the the printer is ready. But this is not a very good way to do it, since you would have to do such PEEKing prior to every print command! The better way is to write a short output routine which checks this bit for itself.

The 6850 occupies two address locations: $F000 and $F001. The first of these is the control register of the 6850 and, by writing and reading this address, one can send and receive control signals. $F001 is the data register and, by writing or reading this address, one can send and receive data from the 6850.

The short output routine shown here illustrates how one might check for a printer busy signal. The listing includes two small programs that turn the RTS signal off and on. The latter might be employed to write a SAVE routine that could be called from BASIC and would turn the cassette or printer on automatically. Remember that you will have to put the addresses of your I/O routines in locations $0218, $021A, $021E and $0220 after each time you depress the BREAK key.

The J3 Port

The main purpose for J3 is to interface peripherals which require RS-232 signals. As can be seen in Figure 1, pins 2 and 3 are the data out and in pins. Pin 7 provides a negative voltage for the RS-232 interface. To use this, however, you will have to open the ground at jumper W10, the lower one under Q1. Even more than this, you will have to install all the hardware for the RS-232 signal level generation; that is, Q1 and Q2 and their associated resistors and diode. Once again you must set up W10 and W11 with the proper switch, as described previously, so that you can switch between the cassette and your peripheral. I believe that the description for J2 was sufficient to get you going on the software you might need to use this port.

The J4 Port

In the OSI manual on the Superboard, J4 is described as a "joystick" and "noise" port. The noise is made by turning on and off four of the keyboard

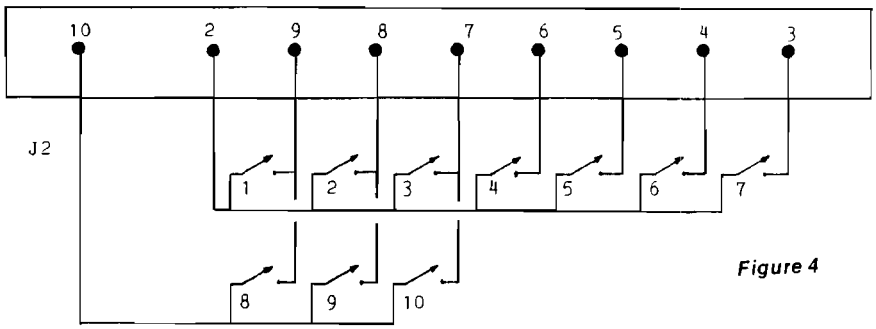


Figure 4

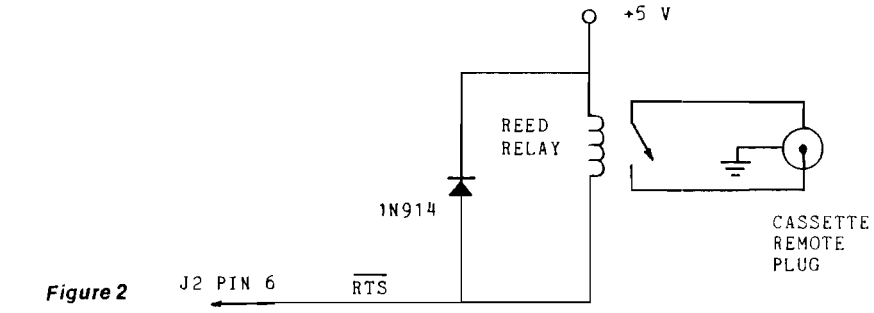


Figure 2

Figure 1: Superboard I/O Ports

J1

Pin	Signal
1	$\overline{IRQ}$
2	NMI
3	DD
4	BD0
5	BD1
6	BD2
7	BD3
8	GND
9	GND
10	GND
11	unused
12	A2
13	A1
14	A0
15	A3
16	A4
17	A5
18	A6
19	A7
20	A8
21	A9
22	A10
23	A11
24	A12
25	A13
26	A14
27	A15
28	GND
29	GND
30	GND
31	02
32	R/W
33	BD7
34	BD6
35	BD5
36	BD4
37	GND
38	GND
39	GND
40	GND

J2

Pin	Signal
1	RxData
2	RxCLK
3	$\overline{CTS}$
4	TxData
5	TxCLK
6	$\overline{RTS}$
7	Mic .05 volt
8	GND
9	AUX 0.5 volt
10	Audio in
11	GND
12	Video out

J3

Pin	Signal
1	GND
2	RS232 out
3	RS232 in
4	RxData
5	RxData1
6	RxData2
7	-V in for RS232 inte·face
8	unused
9	$\overline{CTS}$
10	$\overline{CTS2}$
11	unused
12	unused

J4

Pin	Signal
1	R1
2	R7
3	C1
4	C2
5	C3
6	4
7	C5
8	C6
9	C7
10	R6
11	GND
12	Noise

latches. These are coupled through resistors and a capacitor to pin 12 of J4. The main problem is that the resistors are not installed, nor are their values given. I have not experimented enough with these to determine what values would work best to give four bit analog output.

The main reason I have not done this experimentation is that I have not thought the "noise" would be very useful because it is coupled to the keyboard. For this reason, whenever the keyboard input routine is called, a tone is generated by a loop in that routine which sets and resets the keyboard latches.

If you wanted to produce some music, you could do so by choosing proper values for these resistors and then writing a small program to turn on and off these latches by writing to address $DF00. I would advise installing a switch between the output of pin 12 and your amplifier since you will want to turn off this noise whenever you are not generating some music or gaming sound effects. The keyboard routine's continuous tone is rather annoying after a while!

If you want a beeper to signal various conditions audibly, then I recommend that you use the RTS output at J2. It comes from a heavy buffer which could be connected through a 100 ohm resistor and a small speaker to the 5 volt line. When this RTS signal is turned on and off at the proper rate, it would make a nice beeper without the need for the amplifier that the output at J4 pin 12 requires. Also, there would be no annoying continuous tone.

The other pins on J4 are quite useful because they are connected directly to the keyboard matrix. The graphics manual has a short description of how to deactivate the CTRL-C routine and how to check for a key depressed. If you were to connect lines 1 through 11 on J4 to some switches, you could use the procedure to determine whether the switches were closed. In this way, one might simulate a joystick.

By using four switches you could indicate eight directions. North, east, south and west could be indicated when exactly one switch was closed — the switch in that particular direction on your joystick. Northeast, southeast, southwest and northwest could be indicated by two adjacent switches being closed at the same time. By this means you could move a point on the screen in any of eight directions.

Another very good use for these lines would be to add a numeric keypad in parallel with the keyboard. To do so, you need only wire the switches on the keypad so they are in parallel with the corresponding keys on the keyboard as shown in the schematic, sheet 12. See

Figure 4 for a diagram of these switches. By doing this and writing a short BASIC program, you could imitate a very powerful calculator.

The J1 Port

This port is what OSI uses for expansion. It has all the data and address lines in addition to several of the control lines that the 6502 produces. I suggested in my previous article that this socket could be connected to a KIM type connector to make a KIM expansion port. That is more or less true but, as you will see from checking the signals available on J1 and the required signals on the KIM expansion port, there are a few missing. The most important ones are there, and it just may be that the ones you need to operate your peripheral memory board or whatever are present.

Pin 3, the DD line, needs some explanation. This line is an incoming signal that is used to control the data buffers U6 and U7. This line must be driven by the $\overline{R/W}$ signal, so I suggest that you connect both the $\overline{R/W}$ signal (that you get from U21 pin 6) and the line from J1 pin 3 to the $\overline{R/W}$ pin on the KIM expansion connector.

I think a 40 wire ribbon cable with a DIP plug on the end of it would be the best thing to make the connection from J1 to the KIM connector. Of course, some of the wires won't be used; and so, you might be able to pull some of the unused wires out and solder them to the points on the Superboard where you are going to get the missing signals.

The missing signals can be found at the following places: $\overline{R/W}$ on U21 pin 6 as mentioned above, $\overline{02}$ on U21 pin 4, $\overline{RST}$ on the high (non-ground) side of the BREAK key, VCC where the red 5 volt supply line enters the board, VSS any place along the edge of the board where the ground plane is, SYNC on J8 pin 7, and 01 on U8 pin 3. If you need the RDY signal, you have to make a change on the Superboard. Open the short trace coming from U8 pin 2, which is the RDY line on the 6502, and put a 4.7K pull up resistor in the opening you have made. This will enable any peripheral that needs to use the RDY line to pull it low. After installing the resistor, you can wire the RDY line to U8 pin 2.

There are also R0, K6, SST OUT, RAM/R/W, and PLL TEST lines on the KIM expansion connector, but you won't be able to get these from the Superboard. I doubt that any of the peripherals you might be interested in will require them since they are rather peculiar to the KIM.

This method of directly wiring a KIM socket to the appropriate signals on the Superboard will give you a workable KIM expansion connector even though it may look a little messy since you have to run wires to several points on the Superboard. If you plan to use several boards simultaneously, you will want to make your connections to a KIM compatible motherboard.

You may ask if all this wiring is worth the effort, since OSI sells a 610 expander board which plugs directly into the J1 socket and which will then connect to the OSI 48-pin bus. I think that it is because I like to work with hardware and software together. OSI doesn't offer everything that I need, and their price is somewhat high for what I want. You may wish to investigate just what OSI offers in the way of peripherals before you make any of these changes and additions to your Superboard. In any case, I hope that you now understand a little more about how your Superboard works and how you might go about connecting some peripherals to it.

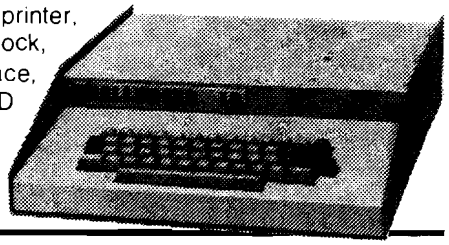

```
PRINT   CRG     (Wherever you want it)
CRT     EQU     $BF2D
STATUS  EQU     $F000
DATA    EQU     $F001
SAVFLG  EQU     $0205
        JSR     CRT     OUTPUT TO CRT
        PHA             SAVE CHARACTER
        LDA     SAVFLG  CHECK SAVE FLAG
        BEQ     RTN     IF 0 NO 6850 OUTPUT
WAIT    LDA     STATUS  WAIT FOR
        LSRA              CHARACTER
        LSRA              TO BE TRANSMITTED
        BCC     WAIT
WAIT1   LDA     STATUS  WAIT FOR
        ANDIM   $08       PRINTER
        BNE     WAIT1     READY
READY   PLA             WHEN READY
        STA     DATA      OUTPUT DATA
RTN     RTS

CASOFF  LDAIM $51
        STA     STATUS
        RTS

CASON   LDAIM   $11
        STA     STATUS
        RTS
```

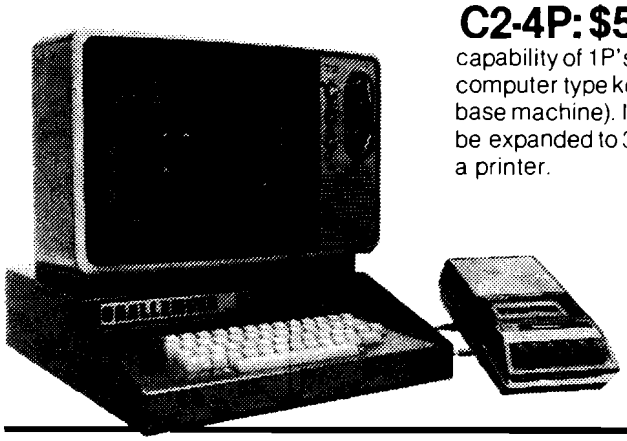
Figure 3

WE'VE GOT YOUR COMPUTER

C1P: $349! A dramatic breakthrough in price and performance. Features OSI's ultra-fast BASIC-in-ROM, full graphics display capability, and large library of software on cassette and disk, including entertainment programs, personal finance, small business, and home applications. It's a complete programmable computer system ready to go. Just plug-in a video monitor or TV through an RF converter, and be up and running. 15K total memory including 8K BASIC and 4K RAM — expandable to 8K.

C1P MF: $995! First floppy disk based computer for under $1000! Same great features as the C1P plus more memory and instant program and data retrieval. Can be expanded to 32K static RAM and a second mini-floppy. It also supports a printer, modem, real time clock, and AC remote interface, as well as OS-65D V3.0 development disk operating system.



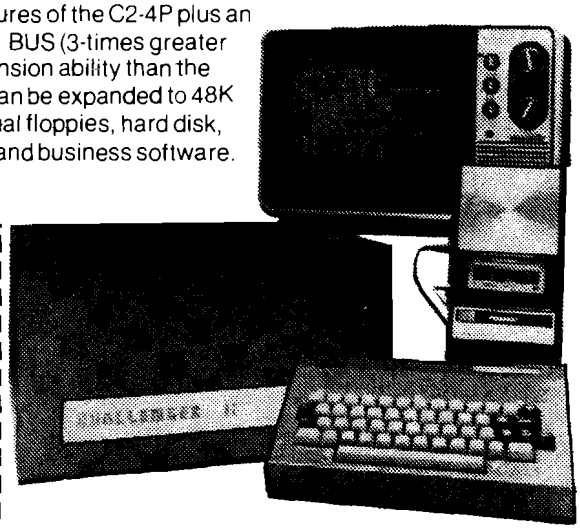


C2-4P: $598! The professional portable that has over 3-times the display capability of 1P's. Features 32 x 64 character display capability, graphics, full computer type keyboard, audio cassette port, and 4 slot BUS (only two used in base machine). It has 8K BASIC, 4K RAM, and can be expanded to 32K RAM, dual mini-floppies and a printer.

C2-4P MF: $1533! It's a big personal computing mini-floppy system at a special package price. Contains the famous C2-4P microcomputer with 20K static RAM, 5" mini-floppy unit for instant program and data loading, RS-232 circuitry (for optional modem and printer), and diskettes featuring exciting games, personal, business and education applications.

C2-8P DF: $2599! A full business system available at a personal computer price! The system includes the powerful C2-8P microcomputer (32K RAM expandable to 48K), dual 8" floppy unit (stores 8-times as much information as a mini-floppy), and 3 disks of personal, educational and small business applications software. Has all the capabilities of a personal system including graphics plus the ability to perform Accounting, Information Management, and Word Processing tasks for small business.

C2-8P: $799! The personal class computer that can be expanded to a full business system. Has all the features of the C2-4P plus an 8 slot BUS (3-times greater expansion ability than the C2-4P). Can be expanded to 48K RAM, dual floppies, hard disk, printer and business software.



I'm interested in OSI Computers. Send me information on:

☐ Personal Computers ☐ Small Business Computers
☐ Educational Systems ☐ Industrial Development Systems

Name _____

Address _____

City _____

State _____ Zip _____

Phone _____

COMPUTERSHOP

Boston	Union N H	Cambridge
590 Comm. Ave. (across from B.U.) 247-0700	Rte 16B 603-473-2323	288 Norfolk St. (near M.I.T.) 661-2670

PROGRESSIVE SOFTWARE

Presents
Software and Hardware for your APPLE

SALES FORECAST provides the best forecast using the four most popular forecasting techniques: linear regression, log trend, power curve trend, and exponential smoothing. Neil D. Lipson's program uses artificial intelligence to determine the best fit and displays all results for manual intervention. **$9.95**

CURVE FIT accepts any number of data points, distributed in any fassion, and fits a curve to the set of points using log curve fit, exponential curve fit, least squares, or a power curve fit. It will compute the best fit or employ a specific type of fit, and display a graph of the result. By Dave Garson. **$9.95**

PERPETUAL CALENDAR may be used with or without a printer. Apart from the usual calendar functions, it computes the number of days between any two dates and displays successive months in response to a single keystroke. Written by Ed Hanley. **$9.95**

STARWARS is Bob Bishop's version of the original and best game of intergallactic combat. You fire on the invader after aligning his fighter in your crosshairs. This is a high resolution game, in full color, that uses the paddles. **$9.95**

ROCKET PILOT is an exciting game that simulates blasting off in a rocket ship. The rocket actually accelerates you up and over a mountain; but if you are not careful, you will run out of sky. Bob Bishop's program changes the contour of the land every time you play the game. **$9.95**

SPACE MAZE puts you in control of a rocket ship that you must steer out of a maze using paddles or a joystick. It is a real challenge, designed by Bob Bishop using high resolution graphics and full color. **$9.95**

MISSILE ANTI-MISSILE displays a target on the screen and a three dimensional map of the United States. A hostile submarine appears and launches a pre-emptive nuclear attack controlled by paddle 1. As soon as the hostile missile is fired, the U.S. launches its anti-missile controlled by paddle 0. Dave Moteles' program offers high resolution and many levels of play. **$9.95**

MORSE CODE helps you learn telegraphy by entering letters, words or sentences, in English, which are plotted on the screen using dots and dashes. Ed Hanley's program also generates sounds to match the screen display, at several transmission speed levels. **$9.95**

POLAR COORDINATE PLOT is a high resolution graphics routine that displays five classic polar plots and also permits the operator to enter his own equation. Dave Moteles' program will plot the equation on a scaled grid and then flash a table of data points required to construct a similar plot on paper. **$9.95**

UTILITY PACK 1 combines four versatile programs by Vince Corsetti, for any memory configuration.

- **Integer to Applesoft conversion:** Encounter only those syntax errors unique to Applesoft after using this program to convert any Integer BASIC source.
- **Disk Append:** Merge any two Integer BASIC sources into a single program on disk.
- **Integer BASIC copy:** Replicate an Integer BASIC program from one disk to another, as often as required, with a single keystroke.
- **Applesoft Update:** Modify Applesoft on the disk to eliminate the heading always produced when it is first run.
- **Binary Copy:** Automatically determines the length and starting address of a program while copying its binary file from one disk to another in response to a single keystroke. **$9.95**

BLOCKADE lets two players compete by building walls to obstruct each other. An exciting game written in Integer BASIC by Vince Corsetti. **$9.95**

TABLE GENERATOR forms shape tables with ease from directional vectors and adds additional information such as starting address, length and position of each shape. Murray Summers' Applesoft program will save the shape table anywhere in usable memory. **$9.95**

OTHELLO may be played by one or two players and is similar to chess in strategy. Once a piece has been played, its color may be reversed many times, and there are also sudden reverses of luck. You can win with a single move. Vince Corsetti's program does all the work of keeping board details and flipping pieces. **$9.95**

SINGLE DRIVE COPY is a special utility program, written by Vince Corsetti in Integer BASIC, that will copy a diskette using only one drive. It is supplied on tape and should be loaded onto a diskette. It automatically adjusts for APPLE memory size and should be used with DOS 3.2. **$19.95**

SAUCER INVASION lets you defend the empire by shooting down a flying saucer. You control your position with the paddle while firing your missile at the invader. Written by Bob Bishop. **$9.95**

HARDWARE

LIGHT PEN with seven supporting routines. The light meter takes intensity readings every fraction of a second from 0 to 588. The light graph generates a display of light intensity on the screen. The light pen connects points that have been drawn on the screen, in low or high resolution, and displays their coordinates. A special utility displays any number of points on the screen, for use in menu selection or games, and selects a point when the light pen touches it. The package includes a light pen calculator and light pen TIC TAC TOE. Neil D. Lipson's programs use artificial intelligence and are not confused by outside light. The hi-res light pen, only, requires 48K and ROM card. **$34.95**

POSTAGE AND HANDLING

Please add $1.00 for the first item
and $.50 for each additional item.

- **Programs accepted for publication**
- **Highest royalty paid**

TO ORDER

Send check or money order to:

P.O. Box 273
Plymouth Meeting, PA 19462

PA residents add 6% sales tax.

U.S. and foreign dealer and distributor inquiries invited

All programs require 16K memory unless specified

The MICRO Software Catalog: XIII

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

Name: **Text Processing System** (Editor and Assembler)
System: **APPLE II**
Memory: **24K**
Language: **Integer BASIC and Machine Language**
Hardware: **APPLE II, 24K and Disk II**

Description: This disk based system allows you to create and edit Applesoft, Integer BASIC, assembly language, and APPLE DOS exec files. The text editor provides capabilities to create load, modify and save APPLE II disk operating system text files. Editing features include simple-to-use data entry, extensive character and string searches and replacement, block line movement, and simple single line macros. Text creation and modification is further simplified with such features as tabbing, specific search windows, file merging, and line deletion. The text editor supports systems equipped with a printer to create permanent listings of text files.

The assembler is a complete, disk-based, two pass symbolic assembler. You can assemble up to ten disk based text files at any one time. The assembler will generate disk based binary files that can be executed via the APPLE DOS "BLOAD" or "BRUN" commands. Eight character symbols allow for meaningful variable and routine names. The assembler supports all 56 standard 6502 opcodes and six additional pseudo-opcodes used to define constants, labels, program addresses, etc. Meaningful error messages are generated to help locate program mistakes. The assembler can generate both program and symbol table listings, with optional line printer output.

Copies: **200**

Price: **$55.00 plus $1.00** shipping and handling
California residents add 6 per cent sales tax
Includes: All programs on a diskette and a complete 60-page user's manual.

Author: **Jeffrey Gold**
Available from:
Software Concepts
Box 1112
Cupertino, CA 95014

Name: **Household Finance Program**
System: **APPLE II**
Memory **32K**
Language: **Integer BASIC and Machine Language**
Hardware: **APPLE II, 32K, and Disk II**

Description: The household finance program is a comprehensive household record maintenance and budget management program. This disk based system provides the capability to maintain 175 records a month for 12 months (that's over 2000 records on a single diskette). With a simple to use data entry mode, a user can enter check transactions, deposits, and cash expenditures.

Error correcting is a simple matter with a complete set of editing features. Twelve user definable budget catagories are available to allow a family to plan and analyze spending patterns. Check and cash expenditures can be assigned to any budget category. Both month-to-date and year-to-date budget summaries are available. Additionally, the program will provide data on how well the family is keeping to its established monthly budgets. Previously entered financial records can be retrieved via a comprehensive data listing mode.

Other program features include checkbook balancing, tax deductible classification, and single disk drive copy (backup) to protect against data loss. The program supports systems equipped with a printer and can provide user selected permanent listing via a unique page print mode. This software package is the most complete, easy-to-use home financial program available today.

Copies: **20**

Price: **$3[illegible].95** plus **$1.00** postage and handling
California residents must add 6 per cent sales tax
Includes: All software supplied on a program diskette with a complete 32-page user's manual.

Author: **Jeffrey Gold**

Available from:
Software Concepts
Box 1112
Cupertino, CA 95014

Name: **Belais' Master Index to Computer Programs in BASIC**
System: **All**
Memory: **N/A**
Language: **BASIC** (a few programs require machine language routines)
Hardware: **N/A**

Description: A directory of computer programs written in BASIC. The programs are ones that have appeared in ten major home computer magazines. They cover both business and personal applications. All major computer systems are included. Many of the programs are written specifically to take advantage of the capabilities of such 6502-based computers as the PET and the APPLE II. The reviews provide detailed information about what each program does, and what hardware and software it needs. Program listings are *not* provided, but information is given on where to get them.

Price: **$9.95** plus **$1.00** shipping
California residents must include 6 per cent sales tax.
Includes: Book
Order info: VISA/Master Charge accepted (give account number, expiration date, and signature).

Author: **Paul Belais**
Available from:
Falcon Publishing
Dept. Y
P.O. Box 688
Ben Lomond, CA 95005

Name: **Mailing List Program**
Memory: **48K** with DOS and Applesoft ROM
Language: **Applesoft II**
Hardware: **APPLE II, disk drive, printer**

Description: The mailing list program is a disk based, menu driven program written in Applesoft II. In order to use the program, a 48K system with Applesoft II on firmware along with one disk drive and DOS 3.2 is required. If your system does not have Applesoft II on firmware, the mailing list program can still be used but the number of entries will be greatly reduced.

The program is able to maintain a complete mailing list. The mailing list data base can be changed, sorted, searched, added, deleted and reformatted. There are five types of sort and five types of search. Labels can be printed out on a 40, 80 or 132 character printer and also viewed on the screen for rapid editing. The program accomodates zip codes with seven digits for use outside the U.S.A.

There is a routine for lining up the labels and for setting the spaces between the labels. Provision has also been made to make a backup copy of the data with a single disk drive. The mailing list program makes generation and maintenance of a mailing list very quick and simple.

Price: **$34.95** for diskette plus $1.25 shipping
Includes: User manual and documentation.

Author: **Gary E. Haffer**
Available from:
Software Technology for Computers
P.O. Box 428
Belmont, MA 02178

Name **Black Box**
System: **APPLE II**
Memory: **16K**
Language: **Integer BASIC**
Hardware: **Cassette**

Description: The program Black Box is based on the Parker Brother's game of the same name. The object of the game is to guess the positions of marbles that are hidden on an eight by eight board. To help you find the marbles, rays are sent into the box. These rays can hit a marble, be deflected by a marble, be absorbed into the box, or any combination of these! There are full instructions inside the program, and a sample game to get you going. Test your reasoning power against the mystical Black Box!

Price: **$8.00**
Includes: Verified cassette, postage and handling

Author: **Robin Hodgson**
Available from:
The AppleCorp
103 Horizon 14
723 14th St. N.W.
Calgary, Alberta
Canada
T2N 2A4

Name: **APPLE—DOC**
System: **APPLE II**
Memory: **3.5 to 5.8K** depending on options.
Language: **Applesoft II**

Description: Set of three programs—VARDOC, LINEDOC, and REPLACE.

VARDOC produces a list of every variable used in your program and all the lines each is used on. Screen and/or printer output can include optional descriptors of each variable.

LINEDOC produces a list of every line called by a GOTO, GOSUB, etc, and all the lines each is called from. You are even alerted to calls to lines no longer in the listing. Optional descriptors are for each line number.

REPLACE allows you to easily rename any or all occurances of any variable in your program. Even change variable types! Can also be used to replace constants or referenced line numbers within the listing. The Literal Mode allows you to replace any set of characters or BASIC statements with any other set. This program is especially useful when appending subroutines with conflicting variable use.

Price: **$9.95** for cassette, **$13.95** for diskette.
California residents must add 6 per cent sales tax.
Includes: Three programs plus documentation.

Author **Roger Wagner**
Available from:
Local Computer Stores or
Southwestern Data Systems
P.O. Box 582
Santee, CA 92071
(714) 562-3670, SASE for free information

Name: **Roger's Easel**
System: **Apple II**
Memory: **16K** for Integer and Applesoft ROM, **20K** for Applesoft RAM

Description: Set of three programs: Roger's Easel, Lo-Res Link-Integer, and Lo-Res Link-Applesoft. A paddle oriented sketching program using the color graphics of the APPLE II. The unique features of this set include the ability to store and retrieve user created pictures from tape or disk, ability to erase with a single keystroke, resuming original color when done, and immediate access to a detailed help list while in the program. The most outstanding feature is the option of permanently linking up to 41 pictures to any Integer or Applesoft program for instant recall at any time. Besides being just plain fun, applications range from putting more creative screen images in your game programs to educational programs for younger children involving shape or color recognition.

Price: **$9.95** on cassette, **$13.95** on diskette
California residents add 6 per cent sales tax
Includes: Three program set with ten-page manual.

Author: **Roger Wagner**
Available from:
Local APPLE dealers or:
Southwestern Data Systems
P.O. Box 582-MC
Santee, CA 92071
(714) 562-3670

Name: **Programmer's Utility Pack**
System: **APPLE II**
Memory: **4K to 6K** (for the prog. itself) depending on the program used.
Language: **Integer and Applesoft**
Hardware: **APPLE II** with cassette or disk drive

Description: Set of 11 programs. Appends, STR$() and VAL() are on printed documentation with the tape version. Programs include: Renumber — Integer & Applesoft, Append — Integer and Applesoft, Line Find — Integer and Applesoft, Address/HEX Converter, Screen Find, Memory Move, and the STR$() and VAL() function simulations for Integer.

By using the various programs one can renumber Integer and Applesoft programs with all GOTO's, etc, being renumbered and the user alerted to unusual situations in the program. These include reference line numbers not in the program, lines referenced by a variable or expression, and a number of others.

Line Find allows the user to locate the actual address range of a line in memory so as to be able to insert CLR, HIMEM:, etc. It can also be used on occasion to recover programs garbaged by dropped bits. Address/HEX Converter converts between the HEX, Integer, and Applesoft address formats. It also provides the two byte breakdown of numbers greater than 256 for use in pointers, etc.

Screen Find is used for printing directly on the screen by POKEing appropriate values into the proper locations in memory. Screen Find gives these values and locations when the characters desired and the horizontal and vertical screen positions are input. Memory Move allows one to move blocks of memory up or down any number of bytes from Integer or Applesoft. The Monitor has a routine similar to this, but it cannot be used to move blocks up a small distance and it is not possible to use it directly from Applesoft.

STR$() simulates the function of this name in Applesoft for use in Integer programs. STR$() in Applesoft converts a number to a string. VAL() is similar but converts strings to numbers.

Copies: **Just Released**
Price: **$16.95.** Calif. residents add 6 per cent sales tax.

Includes: Two cassettes or one diskette plus documentation.

Author: **Roger Wagner**
Available from:
Local Apple dealers, or:
Southwestern Data Systems
P.O. Box 582-MC
Santee, CA 92071
(714) 562-3670

Name: **Softtouch Utility Pac II**
System: **APPLE II**
Memory: **24K** with DOS
Language: **Integer and Applesoft BASIC**
Hardware: **Disk drive**

Description: Set of nine programs on disk. Programs include checkbook update to DOS, update electronic index file, auto-write instructions, find hidden control characters, slow/stop list, disk space, listing headers and exec reader. A complete listing is provided for all programs and programming.

Checkbook update rewrites your original checkbook program for use with the disk drive. Routines have been added to change accounts or list bank names with account numbers, etc. Index update rewrites Bob Bishop's electronic index file for complete automation. A printing routine has been added for hard copy.

Auto write appends subroutines to existing programs, converts integer BASIC listings to Applesoft or vice versa. Auto write documentation gives detailed instructions for using the program to patch in lines in any part of a program or delete illegal lines such as 65535, etc. Find hidden control character displays any control character burried in a catalog name or any listing for both integer or Applesoft BASIC. Disk space is written in Applesoft and gives sectors and bytes left on a diskette. No text files are created by the program and operating time is three seconds. Slow/stop list may be loaded in and used continuously after switching disks or languages. Exec reader will read text files for all of the above with the exception of index file.

Price: **$19.95**
Includes: One diskette plus documentation.

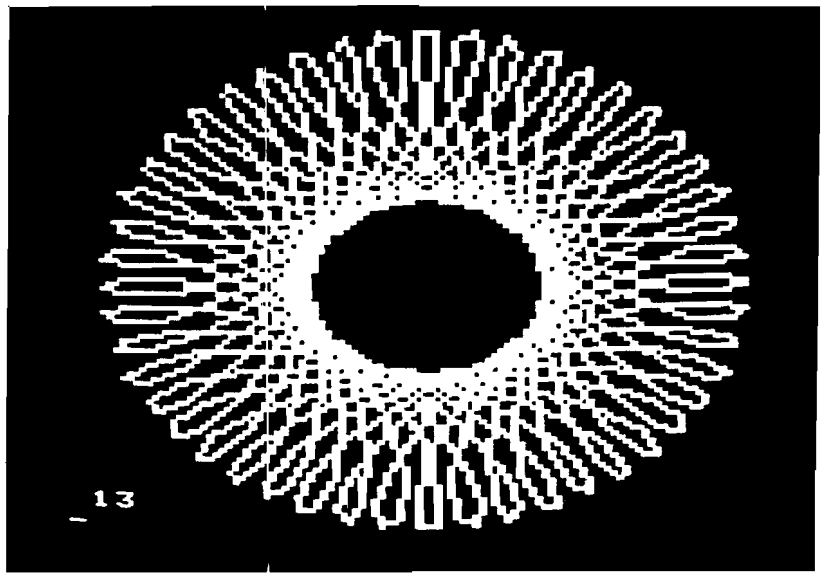
Author: **Dr. Nick Romano**
Available from:
Softtouch
P.O. Box 511
Leominster, MA 01453

Hypocycloids

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

A modification to John Sherburne's original program plots hypocycloids quite a bit faster, on the OSI, by reducing the number of revolutions required. The technique may be used on any micro.

I had just added the extra 2K of memory to my Ohio Scientific 440 video board to implement the graphics option, and was wondering what to do with those 16,384 dots (128 x 128) staring out from my monitor. I happened to pick up the March 79 issue of MICRO and was intrigued by John Sherburne's article on plotting hypocycloids. A hypocycloid, if you don't remember, is what you get when one circle rolls inside another as in the "Spirograph" toy. I immediately accepted the challenge that if it can be done on a PET, I could do it better on my micro.

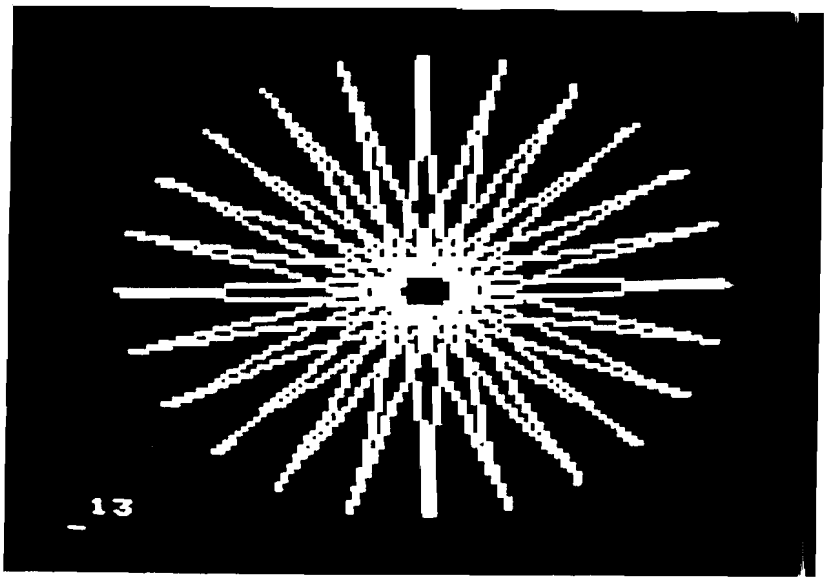



```
ASSEMBLE LIST

                        0100 ;MOVE TBL 1 TO  TBL2
                        0110          .BA    $400
0400— A/  0B            0120 LOOP     LDY    #00
0402— B9  0B  04        0130          LDA    TBL1.Y
0405— 89  0B  05        0140          STA    TBL2.Y
0408— C8                0150          INY
0409  D0  F7              0160          BNE    LOOP
                        0170 ;
040B                    0180 TBL1     .DS    256
050B                    0190 TBL2     .DS    256
                        0200 ;
                        0210                 .EN

LABEL FILE    1 = EXTERNAL

START = 0400        LOOP = 0402        TBL1 = 040B
TBL2 = 050B
110000,060B,060B
```

The original hypocycloid program suffered greatly from lack of speed since each point was calculated using four trigonometric functions. Approximately 300 points per revolution were required. Even then, some gaps appeared in the resulting pattern. I was able to reduce the number of points calculated per revolution to 30 by drawing straight line segments between calculated points. This makes the resulting curves not quite as smooth, but very acceptable as the accompanying photos illustrate. The number appearing in the lower left corner indicates the number of revolutions required to complete the figure.

Below is the subprogram I used to fill in the space between calculated points (X1,Y1) and (X2,Y2). A different procedure is used depending whether the slope of the plotted line is nearer the X axis or Y axis. Lines 1060-1065 and 1160-1165 store the bit in memory and are specific to my graphics board. I would be happy to provide a copy of the full program to anyone who is using the OIS 440 board with graphics.

```
1000 IF X1=X2 THEN 1100
1010 A=(Y2-Y1)/(X2-X1)
1015 IF ABS(A)>1 THEN 1100
1020 B=Y1-A*X1+0.5
1030 FOR X3=X2 TO X1 STEP SGN(X1-X2)
1040 Y3=INT(B+A*X3)
1060 M=54272+16*Y3+INT(X3/8)
1065 POKEM,PEEK(M)ORS(X3AND7)
1070 NEXTX3:RETURN
1100 IF Y1=Y2 THEN RETURN
1110 A=(X2-X1)/(Y2-Y1)
1120 B=X1-A*Y1+0.5
1130 FOR Y3=Y2 TO Y1 STEP SGN(Y1-Y2)
1140 X3=INT(B+A*Y3)
1160 M=54272+16*Y3+INT(X3/8)
1165 POKEM,PEEK(M)ORS(X3AND7)
1170 NEXTY3
1180 RETURN
```

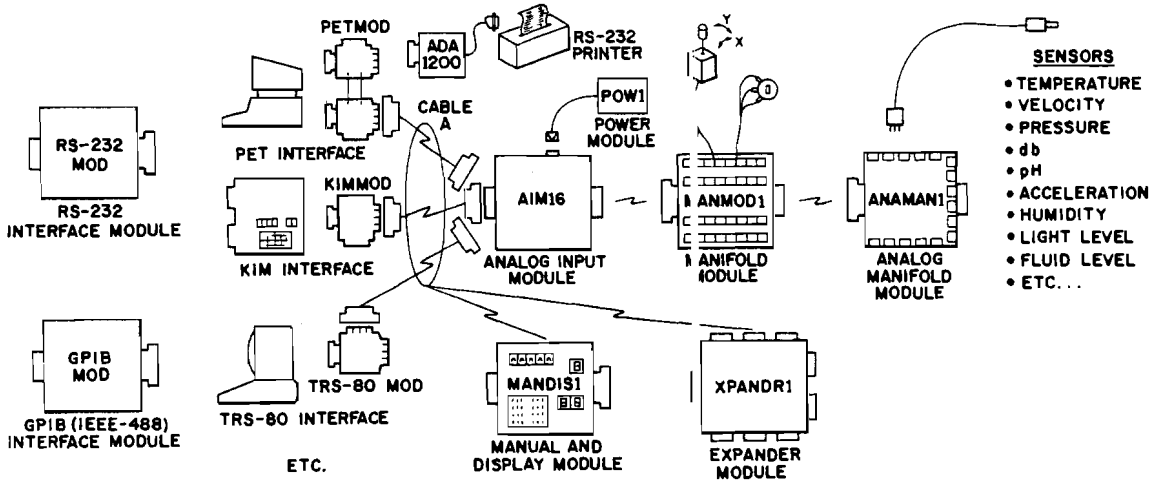


CONNECTICUT microCOMPUTER, Inc.

150 POCONO ROAD - BROOKFIELD, CONNECTICUT 06804

TEL: (203) 775-9659 TWX: 710-456-0052

DAM SYSTEMS



DAM SYSTEMS by CmC
A complete system of modules to let you computer listen to the real world.

DAM SYSTEMS PRICE LIST

DAM SYSTEMS components

AIM16I - Analog Input Module $179.00
16 8-bit analog inputs - 100 microsecond conversion time - 3 state output - requires one 8-bit computer output port for control and one 8-bit computer input port for data.

POW1 - Power Module $14.95
Supplies power for one AIM16 module.

ICON - Input Connector $9.95
For connecting analog inputs to the AIM16 - 20 pin card edge connector - solder eyelets.

OCON - Output Connector $9.95
For connecting the AIM16 to a computer - 20 pin card edge connector - solder eyelets.

MANMOD1 - Manifold Module $59.95
Use in place of ICON. Screw terminal barrier strips for connecting joysticks, potentiometers, voltage sources, etc. Eliminates the need for soldering. Pluss into the AIM16.

ANAMAN1 - Analog Manifold Module TBA
Use in place of ICON. Connects DAM SYSTEMS SENSORS to the AIM16 without soldering - sensor cables just plug in. Pluss into the AIM16 or the MANMOD1.

SENSORS TBA
Sensors for temperature, pressure, flow, humidity, level, pH, motion, etc.

COMPUTER INTERFACES TBA
For the PET, KIM, TRS-80, etc. Use in place of OCON. Eliminates the need for soldering or special construction.

PETMOD - PET Interface Module $49.95
Gives two IEEE ports, one user port and one DAM SYSTEMS interface port. Saves wear and tear on the PET's printed circuit board. Also called the PETSAVR.

KIMMOD - KIM Interface Module $39.95
Gives one application connector port and one DAM SYSTEMS interface port.

CABLE "A" - Interconnect Cables TBA
Connects computer interface to AIM16, MANDIS1, XPANDR1, etc.

CABLE A24 - Interconnect Cable $19.95
24 inch cable with interface connector on one end and an OCON equivalent on the other.

MANDIS1 - Manual and Display Module TBA
Connects between the AIM16 and the computer interface. Allows manual or computer control of the AIM16. Displays channel number and data.

GPIB MOD - GPIB (IEEE-488) Interface TBA
Allows the DAM SYSTEMS MODULES to be used with the GPIB bus instead of a computer's other I/O ports.

RS232 MOD - RS232 Interface Module TBA
Allows the DAM SYSTEMS MODULES to be used with an RS-232 port or terminal.

XPANDR1 - Expander Module TBA
Allows up to 128 8-bit analog inputs (8 AIM16 Modules) to be connected to one system.

DAM SYSTEMS sets

AIM16I Starter Set 1 $189.00
Includes one AIM16I, one POW1, one ICON and one OCON.

AIM16I Starter Set 2 $239.00
Includes one AIM16I, one POW1, one MANMOD1 and one OCON.

PETSET1a $295.00
Includes one PETMOD, one CABLE A24, one AIM16I, one POW1 and one MANMOD1.

KIMSET1a $285.00
Includes one KIMMOD, one CABLE A24, one AIM16I, one POW1 and one MANMOD1.

SYM-1 6532 Programmable Timer

The 6532 interval timer is useful as a backup timekeeper or as a loop controller. It can be accessed in two ways, independent of the interrupt system, and employed to meet a variety of realtime program requirements.

Robert A. Peck
1276 Riesling Terrace
Sunnyvale, CA 94087

In addition to the programmable ports and interval timers located in the 6522s, the SYM-1 has an interval timer in the 6532. The 6532-style device is also used on the KIM-1, and so knowing how to use the SYM timer properly will help in understanding KIM programs and enable the SYM programmer to adapt KIM programs for use on his SYM more easily.

The 6532 timer does not have its IRQ line connected to the IRQ input of the 6502. Therefore, lacking direct access to the interrupt structure, we are unable to get as precise a level of timing as with the onboard 6522s. However, if an extra timer or loop controller is required, the 6532 may prove to be useful.

Before using the timer in the 6532, one must first clear the interrupt flags. Since all of the features we intend to use are part of the write-protected memory, we must first of all allow access to this area. This is accomplished by:

```
20 86 8B      JSR  ACCESS
```

Then, to clear the interrupt flag (PA7 flag), we will read the interrupt flag register. This may be accomplished by reading any one of four locations: A405, A407, A41D or A41F, typically by executing the instruction:

```
AD 05 A4      LDA  INTREG
```

After this instruction is executed, the interrupt flag register will contain "80". This register will be cleared to "00" when we write a value into the timer register. We may then go back occasionally during program execution, test to see if the flag register is still zero, and branch if it is not zero.

As another alternative, we can do a BIT test on the flag location, checking only the timer flag for the branch condition. This method has been used in the sample program. If the BIT test is used, it is not necessry to read the interrupt register in order to clear the PA7 flag because this flag will not be tested. The initial read instruction then becomes redundant.

At this point, we must decide how many clock cycles are to elapse before the timer flag becomes set. Then we will write the selected value into the counter.

There are four different points at which to enter data into the counter, A41C, A41D, A41E and A41F. These are indicated in the manual as 1T 8T, 64T and 1024T. These multiples mean that any data which is entered nto the counter will begin at that particular count and decrement at the rate of the clock frequency (1T), or at one decrement for each eight clock cycles (8T), one decrement for each 64 clock cycles (64T) or one decrement for each 1024 clock cycles (1024T).

There is only one timer register, but the four addresses mentioned above are the means by which the frequency predivider is set. For example, if we write "01" into location A41E, the timer flag is reset and, 64 clock cycles later, the timer flag is set again. If we write "01" into location A41F, instead, then the timer flag will not be reset until 1024 clock cycles have elapsed.

Just as an example, let's say we wanted 800 clock cycles to elapse before the timer flag is set. We will be reading the flag register periodically to see if it is non-zero, determine whether the flag gets set, and branch on the non-zero condition. Writing decimal 00 (hex 64) into location A41D sets the predivider; to 8 then, 8 x 100 = 800 ticks later, the timer reaches zero and the flag is set.

While the counter is independently decrementing, we can determine the current timer contents at any time by reading one of these four locations: A404, A406, A41C, A41E. There are four readable locations due to "don't care" addressing modes or incomplete address decoding.

One might be tempted to look at the timer contents, occasionally, and branch when the count reaches zero. This does not offer a good chance for success as the following example will show.

Let's say we've written "0A" (decimal 10) into location A41D (8T) so that 80 cycles later the timer will count down to zero. Suppose we do the following during the counting period:

(A) Increment a memory location

(B) Test timer contents

(C) Branch back if non-zero

If the sequence of operations takes seven machine cycles, then after 77 cycles the timer will still be at "01" and after 77 + 7 = 84 cycles the timer will contain a count of zero since more than 80 cycles have elapsed, right? Wrong! Unfortunately, it will contain "FC" instead! The limitation of this counter is that, as soon as zero is reached and the flag is set, the counter continues to decrement, but it no longer matters which counter multiple was being used because as the counter immediately begins to free-run decrement at the 1T rate.

To overcome this limitation, since we do not use the IRQ and since we only sample occasionally, we will generally read the interrupt register, testing for a non-zero figure, rather than reading the timer and looking for zero contents as shown above.

Now we come to an example program which ties everything together and demonstrates the use of this timer. Location 20D may be set for any desired timer value. Location 20F may be set to 1C, 1D, 12E, or 1F depending upon whether you want to operate the timer with a predivide of 1T, 8T, 64T, or 1024T. You will notice that the loop of instructions between locations 211 and 224 takes a total of 28 machine cycles to execute.

Begin program execution at location 200. The display will light, upon completion indicating how many times the program was able to traverse the loop before the timer flag became set.

Letters

Having trouble running mnemonically-entered programs on your AIM-65? This might be one source of the problem.

According to the AIM-65 User's Guide, indirect indexed addressing mode may be entered by using either "(HH,Y" or "(HH)Y" where "HH" is a hexadecimal byte. The AIM-65 Summary Card lists the alternatives "(HH,Y" or "(HH,Y)".

However, only the format "(HH)Y" will assemble correctly.

The formats (HH,Y and (HH,Y) will be assembled incorrectly as indexed indirect instructions, "(HH,X)".

Don Stein
6012 Chatsworth Lane
Bethesda, MD 20014


```
                * P OGRAMMABLE TIMER DEMONSTRATION PROGRAM
                *
                * B  ROBERT A. PECK
                * M DIFIED BY MICRO STAFF
                *
0241            ACCI SS *      $8B86
0241            OUTI SP *      $89C1
0241            OUTI YT *      $82FA
0241            SCAI D  *      $8906

0200                    ORG    $0200
0200 A9 00              LDAIM  $00     STORE ZERO IN
0202 85 A0              STA    $00A0   AREA RESERVED FOR TOTAL
0204 85 A1              STA    $00A1
0206 20 86 8B           JSR    ACCESS  UNPROTECT SYSTEM RAM
0209 AD 1F A4           LDA    $A41F   CLEAR PA7 FLAG, OPTIONAL HERE
020C A9 FF              LDAIM  $FF     LOAD TIMER PRESET NUMBER
020E 8D 1D A4           STA    $A41D   ESTABLISH 8 AS PRE-DIVIDE
0211 F8         TMIN    SED            TIME = 255 * 8T = 2040 CYCLES
0212 A5 A0              LDA    $00A0   SET DECIMAL MODE
0214 69 01              ADCIM  $01     LOAD A0 AND ADD ONE
0216 85 A0              STA    $00A0   PUT IT BACK
0218 A5 A1              LDA    $00A1   IF THERE'S A CARRY
021A 69 00              ADCIM  $00     ADD IT IN
021C 85 A1              STA    $00A1   AND RESTORE
021E D8                 CLD            CLEAR DECIMAL MODE
021F 2C 05 A4           BIT    $A405   TEST TIMER FLAG
0222 30 03              BMI    TMOUT   BRANCH IF MINUS FLAG IS SET
0224 4C 11 02           JMP    TMIN    JUMP BACK AND DO IT AGAIN
0227 A9 20      TMOU'   LDAIM  $20     ASCII BLANK
0229 20 C1 89           JSR    OUTDSP  SEND IT TO DISBUF
022C A5 A1              LDA    $00A1   GET CONTENTS OF A1
022E 20 FA 82           JSR    OUTBYT  SEND IT TO DISBUF
0231 A5 A0              LDA    $00A0   NOW GET A0
0233 20 FA 82           JSR    OUTBYT
0236 A9 20              LDAIM  $20     ASCII BLANK
0238 20 C1 89           JSR    OUTDSP
023B 20 06 89   DSCAI   JSR    SCAND   SCAN THE DISPLAY
023E 4C 3B 02           JMP    DSCAN   DO IT CONTINUOUSLY

SYMBOL TABLE 2000 2( 2A
ACCESS 8B86     DSCAI  023B    OUTBYT 82FA    OUTDSP 89C1
SCAND  8906     TMIN   0211    TMOUT  0227
```

While working on a lea ing rate calculation program in Kim BASIC I found the need for a list of variables available so that I could cros s out the ones I used in my program. I fc und such a list in MICRO 4:4 and decide I to write a program, in BASIC, to print it when needed.

Henri Relher
4236 Madison
Montreal, QUEBEC
CANADA H4B 2T9

```
100 REM PROG TO SHOW NUMERICAL AND STRING VARIABLES AVAILABLE IN
110 REM MICROSOFT BASIC AS USED IN PET-APPLE-TRS80 AND OTHERS
115 REM REF:  M I C R O  APRIL-MAY 78 PAGE 4:4
120 FOR X = 65 TO 90
125 PRINT
130 PRINT CHR(X);" ";
140 FOR Y = 0 TO 9
145 Y$ = CHR(X) + NUM(Y) + " "
147 REM INSTEAD OF NUM(Y) YOU CAN USE STR$(Y)
150 PRINT Y$;
155 NEXT Y
160 FOR Z = 65 TO 90
170 PRINT CHR(X);CHR(Z);" ";
180 NEXT Z
200 PRINT
210 NEXT X
220 END
```

More LETTERS

I have a SYM-1. While debugging a program that uses the timer in the 6532 I found out that the IRQ pin is not connected to the IRQ bus. Rather than spend a lot of time finding the neatest way to connect the 6532 IRQ pin to the IRQ bus, I simply ran a piece of wire wrap stock between the IRQ pin on the 6532 to the nearby 6522. Now I can use the interrupt feature of the 6532. I do not know whether Synertek did this for a particular reason but I have not had any problems since making this little modification. Perhaps you are already aware of this. I just thought I would pass it along, for what it is worth.

Keith Le Baron
1260 S. Blackhawk
Freeport, IL 61032

There is a useful, but unadvertised, display subroutine in the AIM-65 Monitor. It is labeled OUTDD1, and can be called by a JSR instruction to hex address EF7B.

The subroutine displays the ASCII character which is in the accumulator, at the relative position (0 - 19 decimal, or 0 - 13 hexadecimal) indicated by the X register. It returns with A and X contents intact.

Before calling the subroutine, be sure to ORA #80, or else the hardware cursor will be displayed.

Don Stein
6012 Chatsworth Lane
Bethesda, MD 20014

[Editor's Note: Marvin De Jong demonstrated the use of this subroutine in an earlier issue of MICRO. Since, however, Don Stein independently "found" it and thinks that it is important enough to point out to other AIM users, we are printing his letter.]

[Editor's Note: If you have some small bit (byte?) of information that you wish to pass on to fellow computerists, a short letter to MICRO is one simple way to "pass the word along".]

A Real-Time Clock for OSI Disk Systems

Did you know that your OSI disk-based system has most of the hardware you need for a realtime clock already built in? Here is information on how to use it.

Robert T. Kintz
104 Council Rock Avenue
Rochester, NY 14610

For most personal and business applications, the need for keeping track of time is either not very great or can be handled by special software routines for particular applications. Where microcomputers are involved in process control operations, however, such as in the real-time control of laboratory experiments, precise timekeeping is a must. Here the initiation and sequencing of most computer-controlled events must be held in tight lock-step with a real time clock.

Owners of Ohio Scientific Challenger II and III disk-based systems may not be aware that provision for a real-time clock already exists on their 470 disk controller board. The bottom middle section of this board contains the PC foils to mount three 74390 decade counter IC's. These divide the on-board 1 MHz crystal clock to provide pulses ranging from 1 to 100,000 per second, selectable at the user's option.

Timing pulses may be fed into the NMI or IRQ lines of the OSI bus (pins 2 or 3) where the 6502 will see them as interrupt signals. The software to handle an interrupt-driven, time keeping routine must have been loaded into memory prior to turning the clock on, or it may be permanently located in PROM at a convenient memory address.

One example of how the hardware may be implemented is shown in Figure 1. A 0.1 Hz clock pulse from the third 74390 is fed into both inputs of a two-input nand gate (7400) after passing through a switch located on the front panel. The 7400 may be conveniently located in the prototyping area just below the three 74390's on the 470 board.

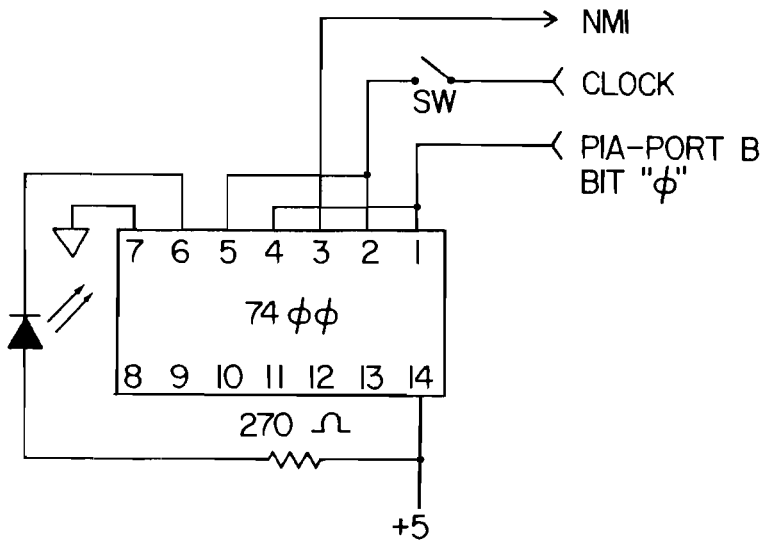


Figure 1

The second input to the two nand gates is taken from bit "0" of a 6821 PIA located on the 500 or 510 CPU board. The outputs of the two 7400 gates are fed to the NMI bus line and a front panel LED, respectively. The brightly flashing LED serves as a reminder that the clock is running, following turning the switch "on" and setting bit "0" high.

The actual interrupt handling and clock routines have been written in machine language, as shown, where they have been assembled to start at $6900 (26880). Of course, relocation of these routines, as well as the clock counters, is entirely optional. Be sure, however, that they are located above the workspace occupied by BASIC or other applications programs.

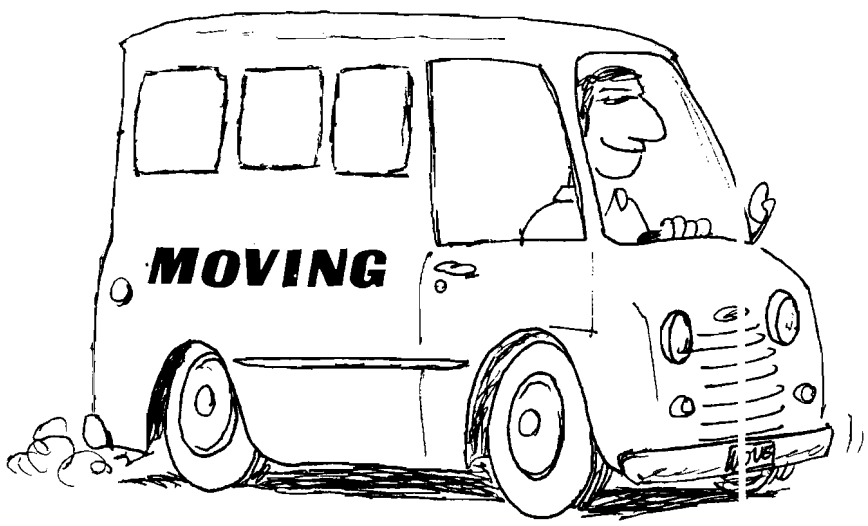
A BASIC demonstration program incorporating the clock is also shown. Lines 50-70 set up the PIA on the CPU board (63232) so that ports A and B are configured as inputs and outputs, respectively. Since OSI's PROM monitor vectors to $0130 on receipt of an NMI interrupt, lines 90-100 POKE a jump to the start of the interrupt handling routine.

Next, in lines 120-140, the machine language object code is read as data and POKEd into high memory. The decimal equivalents of the object code are represented as DATA in lines 9010-9110. Lines 200-220 now set the clock counter locations to "0" and we are ready to turn the clock switch "ON".

Once this is accomplished, the clock is under program and/or keyboard control via POKEs to the PIA PORT B, bit "0." Applications programs inserted at line 300 may use the clock by PEEKing at the appropriate clock counter locations.

```
6957            HOURS   *       $6978
6957            MIN     *       $6979
6957            SECS    *       $697A
6957            FSEC    *       $697B
6900                    ORG     $6900
6900 48         START   PHA
6901 8A                 TXA
6902 48                 PHA
6903 98                 TYA
6904 48                 PHA
6905 20 0E 69           JSR     CLOCK
6908 68                 PLA
6909 A8                 TAY
690A 68                 PLA
690B AA                 TAX
690C 68                 PLA
690D 40                 RTI
690E 78         CLOCK   SEI
690F F8                 SED
6910 18                 CLC
6911 AD 7B 69           LDA     FSEC
6914 69 01              ADCIM   $01
6916 8D 7B 69           STA     FSEC
6919 38                 SEC
691A E9 10              SBCIM   $0010
691C D0 36              BNE     END
691E 8D 7B 69           STA     FSEC
6921 AD 7A 69           LDA     SECS
6924 18                 CLC
6925 69 01              ADCIM   $01
6927 8D 7A 69           STA     SECS
692A 38                 SEC
692B E9 60              SBCIM   $0060
692D D0 25              BNE     END
692F 8D 7A 69           STA     SECS
6932 AD 79 69           LDA     MIN
6935 18                 CLC
6936 69 01              ADCIM   $01
6938 8D 79 69           STA     MIN
693B 38                 SEC
693C E9 60              SBCIM   $0060
693E D0 14              BNE     END
6940 8D 79 69           STA     MIN
6943 AD 78 69           LDA     HOURS
6946 18                 CLC
6947 69 01              ADCIM   $01
6949 8D 78 69           STA     HOURS
694C 38                 SEC
694D E9 24              SBCIM   $24
694F D0 03              BNE     END
6951 8D 78 69           STA     HOURS
6954 D8         END     CLD
6955 58                 CLI
6956 60                 RTS
```

```
10 PRIN :PRINT"REAL-TIME CLOCK FOR DISK-BASED OSI SYSTEMS"
20 PRIN :PRINT"ROBERT T. KINTZ, ROCHESTER, NEW YORK"
30 PRIN :PRINT"RUNS UNDER OSI OPERATING SYSTEM OS-65D,V.3.0"
40 REM  ** SET UP PIA;PORT A=INPUT,PORT B=OUTPUT ***********
50 X=63 32:REM PIA ADDRESS OF 500 OR 510 CPU BOARD
60 POKE X+1,0:POKE X+3,0:POKE X,0
70 POKE X+2,255:POKE X+1,4:POKE X+3,4:POKE X+2,0
80 REM  ** SET UP CLOCK ROUTINE ****************************
90 REM  MI VECTORS TO $0130(304)
100 REM $6900(26880)=START OF CLOCK ROUTINE
110 POK  304,76:POKE 305,0:POKE 306,105
120 REM *** READ IN MAC CODE AS DATA ************************
130 FOR CLK=26880 TO 26964
140 REA  MAC:POKE CLK,MAC:NEXT CLK
150 REM *** CLOCK COUNTER LOCATIONS *************************
160 REM $6978(27000)=HOURS
170 REM $6979(27001)=MINUTES
180 REM $697A(27002)=SECONDS
190 REM $697B(27003)=TENTHS
200 REM *** POKE RESET INTO COUNTER LOCATIONS ***************
210 FOR CL=27000 TO 27003
220 POKI  CL,0:NEXT CL
230 REM *** TURN THE CLOCK SWITCH TO 'ON' ******************
240 PRI T:PRINT"TURN THE CLOCK SWITCH TO 'ON'..."
250 PRI T"THEN PRESS 'G','RETURN'":INPUT A$
260 IF  $<>"G" THEN 250
270 REM *** POKE START INTO CLOCK GATE **********************
280 POKI  X+2,1
290 PRI T:PRINT"CLOCK LED SHOULD NOW BE BLINKING"
300 REM ***************************************************
350 REM USER'S PROGRAM CAN BE INSERTED HERE
400 REM TO USE CLOCK, PEEK AT COUNTER LOCATIONS
500 REM ***************************************************
8999 REI *** MAC CODE DATA FOR CLOCK ROUTINE ***************
9010 DATA 72,138,72,152,72,32,14,105
9020 DATA 104,168,104,170,104,64,120
9030 DATA 24,173,123,105,105,1,141,123
9040 DATA 105,56,233,10,208,54,141,123,105
9050 DATA 173,122,105,24,105,1,141,122
9060 DATA 105,56,233,60,208,37,141,122
9070 DATA 105,173,121,105,24,105,1,141
9080 DATA 121,105,56,233,60,208,20,141
9090 DATA 121,105,173,120,105,24,105,1
9100 DATA 141,120,105,56,233,24,208,3
9110 DATA 141,120,105,216,96
9120 REM *** TURN 'OFF' THE CLOCK ***************************
9130 POKI  X+2,0
9140 REM ***************************************************
9999 END
```



MOVING ?

Please notify **MICRO** *of any change of address so that you will not miss any issues. If we receive the change of address information by the 10th of the month, then the next issue of* **MICRO** *will be sent to the new address. We can not be responsible for replacing issues which are missed due to changes of address which you do not send in time. The Post Office does* **NOT** *return the undeliverable copies - so we lose both the postage and the magazine.*

Send change of address to:

MICRO
P.O. Box 6502
Chelmsford, MA 01824

Please include old label or your subscription number.

6502 Bibliography: Part XIII

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

478. The Cider Press 2 No. 1 (April, 1979)

Scribblemonger, John, "FORTH, Ver 1.6", pg. 1.
Forth for the APPLE is 20 times faster than BASIC.

Silverman, Ken, "Computer Terms". pg. 2.
APPLE terms defined and explained.

Nareff, Max J., "Max your APPLE", pg. 2
Another in a series of articles designed to simulate the various MAtriX functions on the APPLE.

Larsen, Leroy W., "Still another BSTAT", pg. 2.
This BSTAT offers choice of hex or decimal and gives you CATALOG so you can enter the name of the program exactly with the cursor and save the program with another cursor move on the APPLE.

Bernard, Phil, "Storing Strings on Tape, or, Is Disk Necessary?", pg. 3.

Anon, "Disk of the Month", pg. 3.
Twenty-five programs on disk.

Vrooman, Gerry, "The APPLE II Memory Map De-Fogger", pg. 4.
Explanation of where various functions are in memory.

Rahl, Robert R., "N-N-N-N-N-N-N-N-N", pg. 6.
Another version of the Hello program for the APPLE disk combining a few new gimmicks.

479. Byte 4 No 4 (April, 1979)

Campbell, Richard, "Cross Pollinating the APPLE II", pgs. 20-25.
A serial I/O port based on an Intel 8251 with RS-232 output.

Zimmerman, Mark, "Simulating Physical Systems — The Two-Dimensional Ideal Gas", pgs. 26-41.
Use your PET to experiment with physical models.

Meushaw, Robert V. "The Standard Data Encryption Algorithm", pgs. 110-126.
Using the KIM-1 in encryption.

480. KB Microcomputing (formerly Kilobaud) No 29 (May, 1979)

Lindsay, Len, "PET-Pourri", pgs. 6-7.
New PET versions of the Microtechnology Unlimited KIM music board and visible memory are in the offing. More on tape head alignment on the PET. A TAPE TEST program from Jim Butterfield is listed.

Anon., "OSI Small Systems Journal", pgs. 8-11
The OSI Small Systems Journal is now published as a section of Microcomputing.

Anon., "New Products", pgs. 14-25.
A new control board for PET, An ADC Adapter module for PET, and Superchip for the APPLE.

Knox, Thomas; Brazil, Ray H.; and Richardson, Robert M. "Letters to the Editor". pgs. 23-24.
Letters discuss advantages and disadvantages of APPLE II and TRS-80.

Pepper, Clement S., "KIMCTR", pgs. 34-38.
This KIM-1 frequency counter/timer can be used with any m cro with comparable features.

481. Southeastern Software Newsletter No 8 (April 1979)

McClelland, Geo., "A Fast Circle Drawing Program", pg. 2.
On the APPLE Use $FDOC, RDKEY. With several examples, a good tutorial. Also explains exclusive OR.

McClelland, Geo., "Program to Print Applesoft Tokens", pg. 4.
Listing of a program to supplement an earlier program to print Integer BASIC tokens.

McC elland, Geo., "Searching for a Small String Embodied in a Larger String", pg. 5.
Simple listing to use with files or data statements, etc.

McClelland, Geo., "Running Disk Programs the Easy Way", pg 6.
Use of the cursor makes reading in those program titles easy.

482. 6502 User Notes No 14 (April 1979)

Zube', Jim, "KIM-1 Banner", pgs. 1-9.
Designed for a 40-column printer.

Larrabee, Robert D., "Check-Out", pgs. 9-14.
How to check out a new program on the KIM without having to continually hit the plus key. Back up feature. And ability to ADD some material in the middle of a program.

Schilling, Heinz Joachim, "BASIC Mod and Programming Hint", pg 12.
A modification to correct a problem of reloading programs on the KIM using Microsoft BASIC.

Grabowsky, Dick, "BASIC Output Paging Mod", pgs 12-13.
How to limit program listing to 16 lines at a time on the KIM using Microsoft BASIC.

McKenna, Sean, "Automatic Line Number Entry Prompt for BASIC", pg 13.
An automatic line numbering input routine for 9-digit KIM BASIC.

Herman, Harvey, "Renumber Addendum and Some Mods", pg. 13.
Hints for KIM Microsoft BASIC.

Grabowsky, Dick, "A New Command for BASIC", pg 15.
Implementing the GET command in KIM BASIC.

Anon, "Computer Language Forum", pg. 17.
Notes and discussions of FOCAL, Tiny BASIC, FORTH and XPLO.

Mackay, A.M., "Accessing the SYM Displays", pg. 18.
A program to output characters on the display.

Kingston, C., "SYM Notes and KIM-4 Compatibility", pg. 18.
Interfacing details for these two units.

Adams, Jim, "Wumpus and Music Box for SYM", pg. 20.
Modifications to implement these two programs on the SYM.

Nelis, Jody, "Manual Corrections", pgs. 20-23.
Corrections for the AIM User's Manual.

Merhar, Milan, "TVT-6 Notes and RAM Expansion", pgs. 24-25.
TVT-6 discussion and a way to fill the lower 4K in KIM.

McCormack, Chris, "Cassette Directory Printout Program", pgs. 25-26.
Prints your tape directory on your TTY or terminal.

483. Stems from Apple 2 No 4 (April, 1979)

Gustafson, Gus, "INT/FP Stop List Program", pg. 4.
BASIC programs for Stop List.

Gustafson, Gus, "Apple Disk Copy Program", pg. 5.
Modified program to permit using two cards and multiple drives.

Sittel, Randy, "Program FRE(0)", pg. 5.
Routines for free bytes no matter what the memory.

484. Circuit News, April 15, 1979

Anon., "Microcomputers Monitor Oil Well Operation"
APPLE II is used in monitoring off shore oil well drilling processes, displaying information continuously on a silent 700 printer and an H-P X-Y plotter.

485. The Pet Gazette, Spring, 1979

Anon., "Beautiful Music", pg 1, 21.
Micro Technology Unlimited is coming out with a PET version of the KIM music board (DAC) and the visible memory.

Butterfield, Jim, "Routines from PET BASIC", pgs. 2-6.
A listing of a large number of routines from PET BASIC.

Anon., "PET Tokens", pg. 8.
A listing of the 255 PET Tokens.

Butterfield, Jim, "Thoughts on PET BASIC", pgs. 10-12.
Hints for PET users, GET statements, the PET timer, precautions for amateur mechanics, print suppression, etc.

Sherman, H., "Machine Language Load Program", pg. 14.
A BASIC program which loads a machine language routine into the PET.

Anon., "Trace", pg. 18.
A machine language program for tracing the progress of a BASIC program.

Strasma, Jim, "Installing a Second Keyboard", pgs. 20-21.
Instructions and discussion of the keyboard installation.

Butterfield, Jim, "Unlist-List Protection", pg. 21.
How to protect your program listing.

Albrecht, Bob and Karl, "PET BASIC for Parents and Teachers", pgs. 24-25.
Part 6 of this continuing tutorial.

Butterfield, Jim, "PET Memory Locations", pgs. 26-28.
Listing of a large number of key locations and functions.

Butterfield, Jim, "Tape Head Alignment", pg 32.
Procedure and program listing of a tape test to help solve this important problem.

486. Design News, April 23, 1979

Stefanides, E.J., "Personal Computers Become Tool of the Average Man", pgs. 42-48.

487. Byte 4 No 5 (May, 1979)

Pfeiffer, Erich A., "Aids for Hand Assembling Programs", pgs. 238-244.
The article's assembly method is used for program development on a KIM-1 microcomputer.

488. Southeastern Software Newsletter Issue No 9 (May, 1979)

Hartley, Tim, "Stop-List", pg. 1.
Stop-List which works with Applesoft.

Hartley, Tim, and McClelland, Geo., "Character Set", pg. 2.
A machine code program to print the entire character set. Also a discussion of how the program works and the use of the disassembler.

Anon., "Applesoft II Merge Program", pg. 3.
For disk or tape versions or ROM version AS II.

Hartley, Tim, "HI-Res Drawing Program", pgs. 4-5.
Written for a disk system with the AS II ROM card but mods are given to change it for use on other combinations.

Anon., "Correcting Disk Files", pgs. 5-7
An addition to the NAMES FILES program given in earlier issues.

489. MICRO No. 12 (May, 1979)

Burnett, Joe, "An AIM 65 User's Notes", pgs. 5-7.
Notes on getting started with the AIM 65.

Carpenter, Chuck, "S-C Assembler II — Super APPLE II Assembler", pgs 9-11.
Machine or assembly language coding is as easy as BASIC with this assembler.

Donato, Joseph, "A PET Hex Dump Program", pgs. 13-15.
Now you can look at your BASIC in ROM or other interesting codes in machine language.

Gierycz, Jack, "Super HI-LO for the SYM-1", pgs. 17-22.
HI-LO with a new twist to the game.

Williams, J.C., "A 100 us 16-Channel Analog-to-Digital Converter for 65XX Microcomputer Systems", pg. 25-29.
How real-time games can be written for the OSI Challenger systems which use a serial terminal run from the ACIA.

Tripp, Robert M., "ASK the Doctor — Part IV. Good News, Bad News", pgs 35-36.
Good news is that only two minor hardware changes improve the high-speed cassette read/write. The KIM read routine is also improved, new uses for the BREAK command are given, and now the register name is displayed during the R command.

Rowe, Mike (Staff), "The MICRO Software Catalog: VIII", pgs 37-38.
Eleven new programs are described.

Doutre, Ben, "Inside the KIM TTY Service", pgs. 39-40.
How to operate the KIM TTY link at 9600 baud.

Kirschner, Frank D., "The Integer BASIC Token System in the APPLE II", pgs. 41-43.
How APPLE stores characters. A meaty article showing how to exercise considerably more control over the BASIC interpreter in your microcomputer.

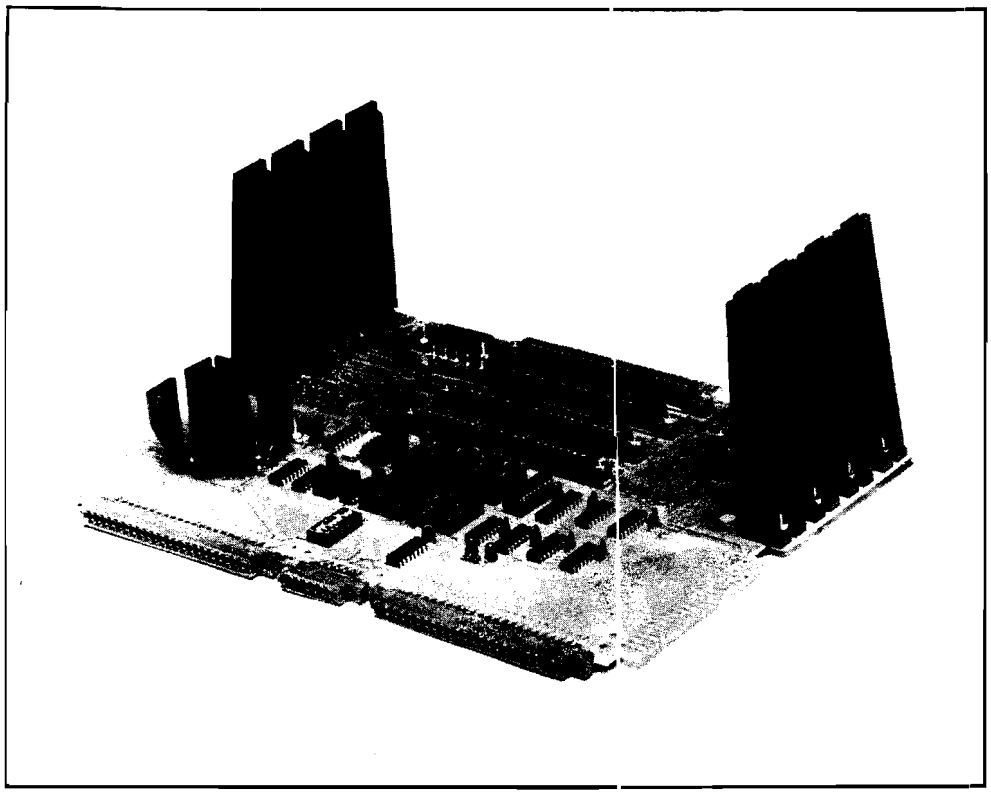
Carpenter, Chuck, "Renumber Applesoft", pgs. 45-46.
Append and renumber routines.

Anon. "Classified Index for Issues 7 to 12", pgs. 47-48.
Index is broken down by system — APPLE, OSI, General, KIM/TIM, SYM/AIM, and so on.

490. 73 Magazine No 12 (May, 1979)

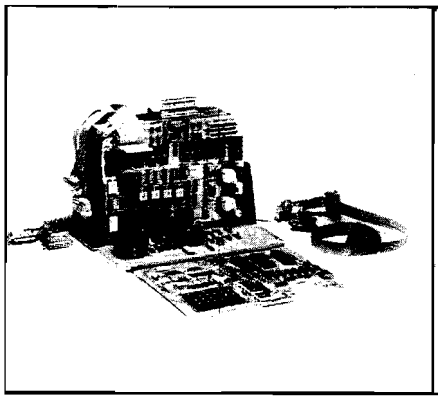
Schmidt, Bill and Shattuck, Bob, "RTTY Transceive for the KIM-1", pgs. 78-82.
This program requires a video terminal and AFSK generator.

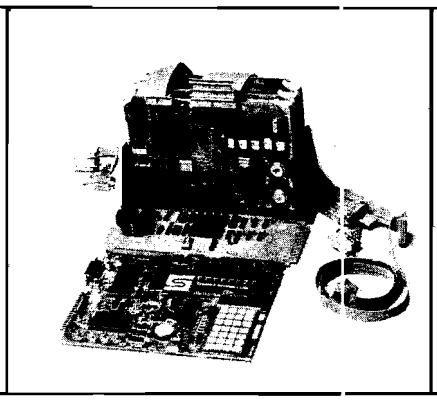
Introducing SEAWELL's

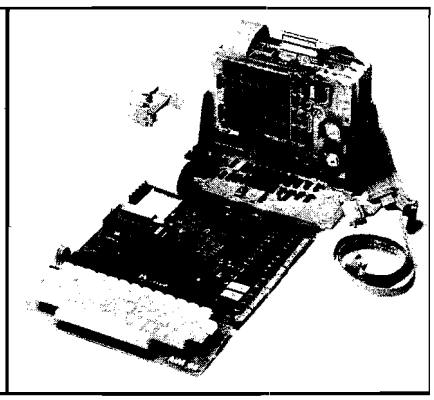


Little Buffered Mother

The ultimate Motherboard for any KIM-1, SYM-1, or AIM-65 system







Features:

- 4K Static RAM on board
- +5V, +12V, and −12V regulators on board
- 4 + 1 buffered expansion slots
- Accepts KIM-4 compatible boards
- Full access to application & expansion connector
- LED indicators for IRQ, NMI, and power-on
- Also compatible with SEA-1, SEA-16, the PROMMER, SEA-PROTO, SEA-ISDC, and more
- Onboc rd hardware for optional use of (128K c ddressing limit)
- Mounts like KIM-4 or with CPU board standing up
- 10 slot Motherboard expansion available - SEAWELL's Maxi Mother

Standard. $139
w/4K RAM. $189
Assembled Only

For further information contact:

SEAWELL Marketing Inc.
P.O. Box 17006
Seattle, WA 98107

SEAWELL Marketing Inc.
315 N.W. 85th
Seattle, WA 98117
(206) 782-9480

POWERSOFT, INC.

P. O. BOX 157
PITMAN, NEW JERSEY 08071
(609) 589-5500

products for the APPLE II

APPLESOFT II UTILITY

(Diskette Only) **$12.45**

The Applesoft II Utility program provides the user with the following features. a) Complete automatic renumbering of any Applesoft II program. b) The creation of an EXEC File for subroutine file creation. This feature allows you to incorporate the same subroutine in various programs. c) No modification of the program in machine memory (RAM). d) Automatic running of the program. No programmer should be without this excellent utility program. REQUIREMENTS: Disk II, Applesoft II, 16K of memory.

REAL ESTATE ANALYSIS PROGRAM

$14.95

The Real Estate Analysis Program provides the user with three features. a) A powerful real estate investment analysis for buy/sell decisions and time to hold decisions for optimal rental/commercial investments. b) Generation of complete amorization schedules consistent with banking practices and schedules. c) Generation of depreciation schedules for selecting the best depreciation schedule for your use and a determination of optimal switch over points to straight line to maximize depreciation. All three features are designed for video screen or printer output. In addition, the program will plot; cash flow before taxes vs. years, cash flow after taxes vs. years, adjusted basis vs. years, capital gains vs. years, pre-tax proceeds vs. years, post-tax proceeds vs. years, and return on investment (%) vs. years. REQUIREMENTS: Applesoft II, 16K of memory without DOS or 32K of memory with DOS (Disk II).

ADDRESS FILE GENERATOR

$19.95

A professional piece of software which allows the user to create four different types of address files: a) Holiday File, b) Birthday File, c) Home Address File, and d) Commercial Address File. The program contains a menu of seven major commands: 1) Create a File, 2) Add to File, 3) Edit File, 4) Display File, 5) Search File, 6) Sort File, and 7) Reorganize File. Most of the major commands have subordinate commands which adds to the flexibility of this powerful software system. We doubt you could buy a better program for maintaining and printing address files. REQUIREMENTS: Disk II, Apple Printer Card, 32K of memory with Applesoft ROM Card or 48K of memory without Applesoft ROM Card.

SUPER CHECKBOOK

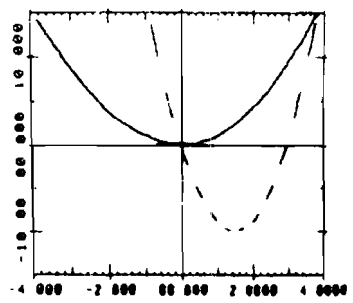
$19.95

A totally new checkbook program with a unique option . . . Bar Graphs. These bar graphs, outputed to a printer or video screen, provide trend analysis data on code expense, income, expenses, or gain/loss on a month by month basis. The program contains a total of fourteen options: 1) Check/Deposit Entry & Modification, 2) Reconciliation of Checks or Deposits, 3) Sort by Check Number, 4) Sort by Code for Year, 5) Sort by Code for Month, 6) Output Year to Date, 7) Output Month Activity, 8-11) Printer/Video Plot Trend Analysis-Bar Graphs, 12) Account Status, 13) Reconciled Check Status, and 14) Quit. An excellent program for maintaining your checkbook, or that of a small business. REQUIREMENTS: Disk II, 32K of memory with Applesoft ROM Card or 48K of memory without Applesoft ROM Card.

FUNCTION GRAPHS AND TRANSFORMATIONS

$14.95

This program uses the Apple II high resolution graphics capabilities to draw detailed graphs of mathematical functions which the user defines in Basic syntax. The graphs appear in a large rectangle whose edges are X and Y scales (with values labeled by up to 6 digits). Graphs can be superimposed, erased, drawn as dashed (rather than solid) curves, and transformed. The transformations available are reflection about an axis, stretching or compressing (change of scale), and sliding (translation). The user can alternate between the graphic display and a text display which lists the available commands and the more recent interactions between user and program. Expected users are engineers, mathematicians, and researchers in the natural and social sciences; in addition, teachers and students can use the program to approach topics in (for example) algebra, trigonometry, and analytic geometry in a visual, intuitive, and experimental way which complements the traditional, primarily symbolic orientation. REQUIREMENTS: 16K of memory with Applesoft ROM Card or 32K of memory without Applesoft ROM Card.



Available at your local computer store

Call or write for our free SOFTWARE & ACCESSORIES CATALOG

DEALER INQUIRIES INVITED

POWERSOFT, INC.

P. O. BOX 157
PITMAN, NEW JERSEY 08071
(609) 589-5500

- Check or Money Order
- Include $1.00 for shipping and handling
- C.O.D. ($1.00 add'tl. charge)
- Master Charge and VISA orders accepted
- New Jersey residents add 5% sales tax

Programs Available on Diskette at $5.00 Additional

softside software

305 Riverside Drive New York, N.Y. 10025

the pet program.

1 **GRAPHICS PAC 2**
New Version

Quadruple your PET's *graphic resolution*. Why be stuck with the PET's cumbersome 25 x 40 1000 point display. With Graphics Pac you can directly control (set and clear) 4000 points on screen. It's great for *graphing, plotting, and gaming*. Graphics Pac allows you to plot in any combination of two modes: 4 Quadrant graphing with (0,0) center screen, and Standard graphing with (0,0) plotted in the upper left hand corner. Complete documentation shows how you can *merge* this useful routine with any of your own programs *without retyping* either one! All this on a high quality Microsette for only $9.95.

2 **ASSEMBLER 2001**

A *full featured assembler* for your PET microcomputer that follows the *standard set of 6502 mnemonics*. Now you can take full advantage of the computing abilities of your PET. *Store* and *load* via tape, *run* through the SYS or USR functions. *List* and *edit* too with this powerful assembler. No other commercial PETassembler gives you all these features plus the ability to *look at* the *PET'S secret Basic* ROMs *all in one program*. This valuable program is offered at $15.95

3 **BIKE**

An *exciting new simulation* that puts you in charge of a bicycle manufacturing empire. Juggle inflation, breakdowns, seasonal sales variations, inventory, workers, prices, machines, and ad campaigns to keep your enterprise in the black. *Bike is dangerously addictive*. Once you start a game you will not want to stop. To allow you to take short rest breaks, Bike lets you store the data from your game on a tape so you can continue where you left off next time you wish to play. Worth a million in fun, we'll offer BIKE at $9.95.

4 **PINBALL**

Dynamic usage of the PET's graphics features when combined with the fun of the *number 1 arcade game* equals an *action packed video spectacle* for your computer. Bumpers, chutes, flippers, free balls gates, a jackpot, and a little luck guarantee a great game for all. $9.95.

Authors: Our royalties are unbeatable

★ ☆☆☆☆☆☆ **MUSICAL MADDNESS** ☆☆☆☆☆☆ ★

SPECIAL add an exciting new dimension to your PET computer **SOUND**
with Soundware's soundsational music box
and sonicsound software from Softside & Soundware

☆THE SOUNDWORKS☆

The *Soundware music box* for your PET comes complete with controllable volume, an earphone jack, a *demo tape* with *two programs*, an instruction book, and a *one year warranty*. this sturdy unit is enclosed in an *attractive* plastic case. Notes tell how to *program your own sound effects*. All this during our *musical madness* for just **29.95**

WORD FUN: *Speller*: fun ways to practice spelling + *Scramble* + *Flashcards* *9.95*

☆MUSICAL SOFTWARE☆

ACTION PACK: *Breakthru* + *Target* + *Catterpillar*: non stop graphic action 9.95

PINBALL: a video action spectacle with real time flippers, chutes gates, bumpers, tags etc. 9.95

CLASSICS: *Checkers* + *Backgammon Board* + *Piano Player*: checkers vs. computer or friend. Piano plays Minute Waltz 9.95

MUSIC MANIA: Try to repeat a growing sequence of tones. With graphics. Challenge to the best ear 9.95

Skyles Electric Works

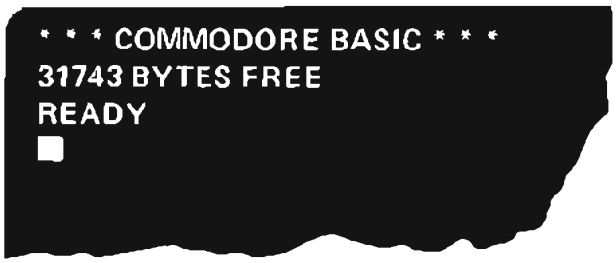
You love your PET, but you'll love it more with this BigKeyboard?

74KB Big KeyBoards @ $125.00 (Plus $5.00 shipping & handling)

The Skyles Big KeyBoard™. More than 15 inches wide. A layout nearly identical to the PET Keyboard and with *all* functions—alpha, numeric, graphics, special symbols, lower case alpha—on full-sized, almost plump, key-tops double-shot to guarantee lifetime durability.

Actual size

Would you like to turn on your PET . . . and see this



8KB 8K Memory Expansion Systems @ $250.00 (Plus $3.50 shipping & handling)

16KB 16K Memory Expansion Systems @ $450.00 (Plus $5.00 shipping & handling)

24KB 24K Memory Expansion Systems @ $650.00 (Plus $5.00 shipping & handling)

Skyles Memory Expansion Systems are complete; nothing more to buy. • First quality static RAMs • Solid soldered on first quality glass epoxy board • Separate PET Adapter Printed Circuit Board connects directly to data bus on your PET—no rat's nest of hanging hand-wiring • Ribbon cable and 50 pin connectors that keep your PET open to the outside world (one on the 8KB; *two* on the 16KB and 24KB).

____ 8KB Memory Expansion System(s) at $250 each. $_____
(Adds 8,192 bytes; total 15,359)(*shipping and handling $3.50 each*)

____ 16KB Memory Expansion System(s) at $450 each. $_____
(Adds 16,384 bytes; total 23,551) (*shipping and handling $5.00 each*)

____ 24KB Memory Expansion System(s) at $650 each. $_____
(Adds 14,576 bytes; total 31,743) (*shipping and handling $7.00 each*)

____ 74KB Big KeyBoard(s) at $125 $_____
(*shipping and handling $5.00 each*)

____ SPECIAL DEAL(S): 8KB Memory and 74KB KeyBoard at $350 complete $_____

____ SPECIAL DEAL(S): 16KB Memory and 74KB KeyBoard at $525 complete $_____

* Please add 6% sales tax if you are a California resident; 6.5% if a resident of BART, Santa Clara or Santa Cruz Counties (CA). Please add shipping and handling costs as indicated.

VISA, MASTERCHARGE ORDERS CALL (800) 227-8398 (except California residents)
CALIFORNIA ORDERS PLEASE CALL (415) 494-1210

Skyles Electric Works

10301 Stonydale Drive
Cupertino, CA 95014
(408) 735-7891