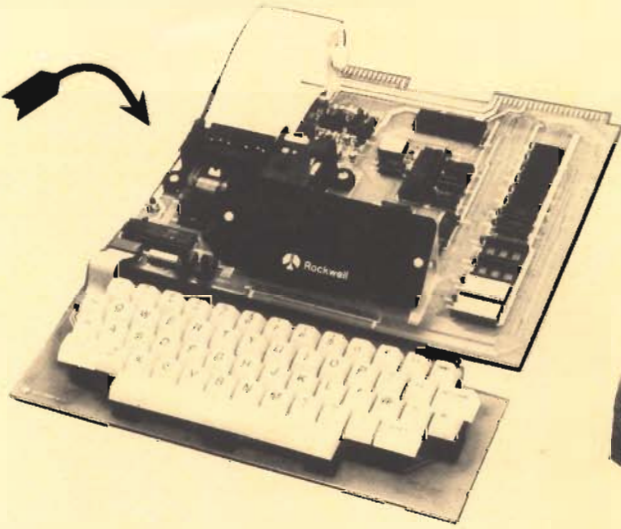


AIM APPLE ATARI KIM OSI PET SYM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
ATARI SYM OSI ATARI AIM PET KIM APPLE SYM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM
PET KIM ATARI SYM OSI ATARI AIM KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
OSI SYM ATARI PET APPLE OSI AIM KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
SYM AIM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
KIM PET APPLE AIM APPLE ATARI KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
APPLE OSI KIM AIM SYM OSI PET ATARI SYM OSI ATARI KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM
AIM APPLE ATARI KIM OSI PET SYM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
ATARI SYM OSI ATARI AIM PET KIM APPLE SYM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM
PET KIM ATARI SYM OSI ATARI AIM PET KIM APPLE AIM APPLE ATARI KIM OSI PET KIM OSI PET SYM APPLE KIM PET AIM
OSI SYM ATARI PET APPLE OSI AIM KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
SYM AIM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
KIM PET APPLE AIM PET ATARI KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
APPLE OSI AIM SYM OSI PET ATARI SYM OSI ATARI AIM PET KIM APPLE AIM AIM SYM PET OSI KIM ATARI APPLE AT
AIM APPLE ATARI KIM OSI PET SYM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
ATARI SYM OSI ATARI AIM PET KIM APPLE SYM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM

MICRO

THE 6502 JOURNAL

Human Physiological Parameters



**LET YOUR
MICRO HELP YOU
GET INTO SHAPE**

AIM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APPLE PET ATARI SYM OSI ATARI A
PET APPLE AIM APPLE ATARI KIM APPLE ATARI KIM OSI PET SYM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APP
LE OSI KIM AIM SYM OSI PET ATARI SYM OSI ATARI AIM PET KIM APPLE AIM AIM SYM PET OSI KIM ATARI APPLE ATARI K
APPLE ATARI KIM OSI PET SYM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APPLE P
RI SYM OSI ATARI AIM PET KIM APPLE SYM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM APP
KIM AIM SYM OSI ATARI AIM PET KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APP
SYM ATARI PET APPLE OSI AIM KIM AIM KIM APPLE PET ATARI SYM OSI ATARI AIM PET KIM APPLE AIM APPLE ATARI C
AIM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APPLE OSI AIM KIM AIM KIM APPLE PET ATARI SYM OSI ATARI A
PET APPLE AIM APPLE ATARI KIM APPLE ATARI KIM OSI PET SYM APPLE KIM PET AIM ATARI OSI SYM ATARI PET APP
LE OSI KIM AIM SYM OSI PET ATARI SYM OSI ATARI AIM PET KIM APPLE AIM AIM SYM PET OSI KIM ATARI APPLE ATARI K

NO. 20

JANUARY 1980

\$2.00

softside software

305 Riverside Drive, New York, N.Y. 10025



SOUNDWARE



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 PET assembler 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

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



SPECIAL

☆☆☆☆☆☆ MUSICAL MADDNESS ☆☆☆☆☆☆

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

SOUND

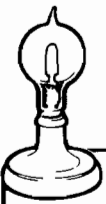
☆ THE SOUNDWORKS ☆

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

The BASIC Programmer's Toolkit

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

Here are Ten Comands you'll need, all on a single chip you can install, in a minute without tools, on any PET or PET system. 2 KB of ROM firmware on a single chip with a collection of machine language programs available to you from the time you turn on your PET to the time you shut it off. No tape to load or to interfere with any running programs.

AUTO DELETE RENUMBER HELP TRACE
STEP OFF APPEND DUMP FIND

```
LIST
10 GOSUB 99
15 PRINT J
16 GOTO 10
99 INPUT J
100 IF J=0 THEN END
200 I = SQR J:RETURN
READY

RENUMBER 100,10

READY
LIST
100 GOSUB 130
110 PRINT I
120 GOTO 100
130 INPUT J
140 IF J=0 THEN END
150 I = SQR(J):RETURN
READY
```

```
RUN
?DIVISION BY ZERO ERROR IN 500
READY
HELP
500 J = SQR(A*B*(%))
READY
```

```
RUN
READY
DUMP
A1 = 10
BW = -6.1
CS = 1 HI
READY
```

```
APPEND "INPUT"
PRESS PLAY ON TAPE #1
OK
SEARCHING FOR INPUT
FOUND INPUT
APPENDING
READY
```

```
TRACE
READY
RUN
ENTER YOUR NAME? JIM
HI JIM
HOW OLD ARE YOU?
#100
#110
#150
#160
#175
#200
```

Can be placed in main board socket or with precision-engineered PCB connector to attach to data bus...depending on the model of your PET and additional memory systems.

Now available to interface

- 8N/8B, 16N/16B, 32N/32B PET...chip only \$50.00*
- 2001-8...chip and interface PCB 80.00*
- With Expandamen, PME 1 80.00*
- R. C. Factor or Skyles Electric Works systems
- With Computhink Disk System 85.00*
- With Commodore's Word Processor II, for original 2001-8 PETs 90.00*
- With Commodore's Word Processor II, for new PETs 72.50*
- With Skyles Macro TeA 50.00*

*Shipping and handling, California sales tax where applicable must be added. See order form attached.

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

VISA, MASTERCARGE ORDERS CALL (800) 538-3083 (except California residents)

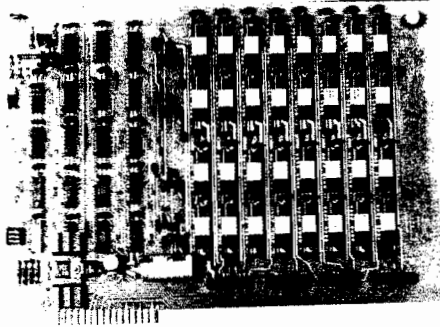
CALIFORNIA ORDERS PLEASE CALL (408) 257-9140



Skyles Electric Works

10301 Stonydale Drive
Cupertino, California 95014
{408}735-7891

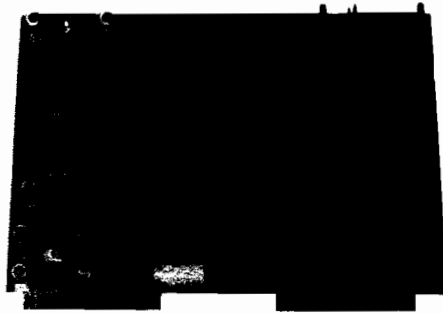
16K MEMORY



K-1016

- ADDRESSED AS CONTIGUOUS 16K STARTING AT ANY 8K BOUNDARY
- LOW POWER — 1.6 WATTS TOTAL
- K-1016A — \$340 6 MONTH WARRANTY

SYSTEM EXPANSION



K-1012

- 12 PROM SOCKETS — 2708/TMS 2716, USES THE POWER OF ONLY 1 PROM.
- 32 BIDIRECTIONAL I/O LINES
- FULL RS-232 ASYNC SERIAL COMMUNICATIONS, 75-4800 BAUD
- PROM PROGRAMMER
- K-1012A — \$295

EXPANSION FOR YOUR 6502 COMPUTER

HIGH RESOLUTION GRAPHICS



- 320 x 200 BIT MAPPED GRAPHICS
- 8K RAM AVAILABLE FOR USE
- EACH POINT INDIVIDUALLY ADDRESSABLE
- K-1008A — \$240, PET — \$243 (PLUS PET INTERFACE)

MULTI-HARMONIC 4 VOICE MUSIC



K-1002-2

MODEL FOR ALL PETS



K-1002

MODEL FOR KIM, AIM, SYM

- FORTIER SYNTHESIZED WAVEFORMS — UP TO 16 HARMONICS
- 4 VOICES PLAY SIMULTANEOUSLY
- QUALITY D/A CONVERTER, 6 POLE FILTER AND AMPLIFIER
- HARDWARE — \$40-50, SOFTWARE — \$20



ALL MTU PRODUCTS ARE SUPPLIED WITH FULL DOCUMENTATION CLASSED AS "BEST IN THE INDUSTRY". MANUALS MAY BE PURCHASED SEPARATELY.



Micro Technology Unlimited
P.O. Box 4596, 841 Galaxy Way
Manchester, N.H. 03108
603-627-1464

Call Or Write For Our Full Line Catalog

Table of Contents

Tape Execute File - Create and Use by Allen J. Lacy	5
Why A PET, APPLE, 6502 BASIC Compiler? by Bruce M. Beach	9
Human Physiological Parameters by Dr. L. S. Feich	15
Lifetime of a Non-Renewable Resource by Marvin L. DeJong	21
Editorial — The Loneliness of the Microcomputer	23
Sweet-16 Programming Using Macros by Richard C. Vile, Jr.	25
Screen Write/File Routine by B. E. Baxter	30
SYM-1 Tape Verification by Jack Gienic	35
Microbes and Miscellanea	39
Symbol Table Sorter/Printer for the AIM Assembler by Mal Evans	43
The MICRO Software Catalogue: XVI by Mike Rowe	51
Search/Change in Applesoft J. D. Childress	55
SYM-1 Staged Loading Technique for Segmented Programs by Robert A. Peck	59
6502 Bibliography: Part XVI by William R. Dial	61

Staff

Editor/Publisher
Robert M. Tripp

Assistant Editors
Mary Ann Curtis
Evelyn M. Heimoch

Business Manager
Maggie E. Elsher

Circulation Manager
Carol A. Stark

Comptroller
Donna M. Tripp

Production Assistant
L. Catherine Bland

MICRO™ is published monthly by MICRO INK, Inc., Chelmsford, MA 01824. Tel. 617/252-6515.

Second Class postage paid at Chelmsford, MA 01824.

Publication Number: 0278-3627

Subscription rates: U.S. - \$15 per year; Foreign, surface mail \$20 per year.

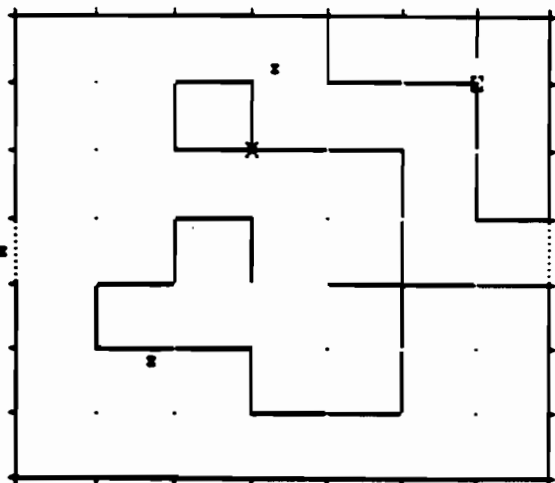
For air mail rate, change of address, back issue or subscription information, write to MICRO, P.O. Box 650, Chelmsford, MA 01824.

Entire contents Copyright © 1979 by MICRO INK, Inc.

Advertiser's Index

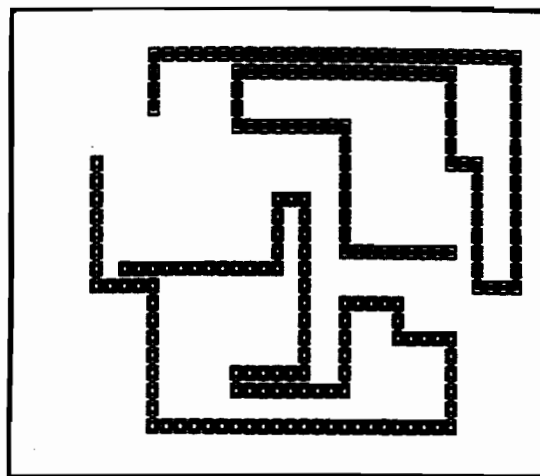
Andromeda Computer Systems	50	Micro Technology Unlimited	254
Apple Shoppe	7	Mighty Byte Computer, Inc.	56
Balkan	56	Muse Software	50
Beta Computer Devices	37	On-Line	50
C & H Micro	48	Perry Peripherals	54
Classified Ads	22,37	Powersoft, Inc.	4
COMPAS	8	Programma International	80
The Computerist, Inc.	13	Progressive Software	53
Computer World	32	Rainbow Computing, Inc.	180
Connecticut microComputers	54	Raygam, Inc.	33
Excent, Inc.	49	RNB-Enterprises	41
Galaxy	54	Shepardson Microsystems, Inc.	38
H. Geller Computer Systems	40	Skyles Electric Works	120
Home Computers	52	Softside Publications	42
Hudson Digital Electronics	34	Softside Software	100
Information Unlimited Software	24	Southwestern Data Systems	58
MICRO	60	S.P.A.R.C.	31
Micro Music, Inc.	14	United Software of America	50

Software for the Apple II



SCORE : 108

DYNAMAZE—a dazzling new real-time game. You move in a rectangular game grid, drawing or erasing walls to reflect balls into your goal (or to deflect them from your opponent's goal). Every ball in your goal is worth 100 points, but you lose a point for each unit of elapsed time and another point for each time unit you are moving. Control the speed with a game paddle: play as fast as ice hockey or as slowly and carefully as chess. Back up and replay any time you want to; it's a reversible game. By Don Stone. Integer Basic (plus machine language); 32 K; \$9.95.



SCORE : 105

ULTRA BLOCKADE—the standard against which other versions have to be compared. Enjoy Blockade's superb combination of fast action (don't be the one who crashes) and strategy (the key is accessible open space—maximize yours while minimizing your opponent's). Play against another person or the computer. New high resolution graphics lets you see how you filled in an area—or use reversibility to review a game in slow motion (or at top speed, if that's your style). This is a game that you won't soon get bored with! By Don Stone. Integer Basic (plus machine language); 32 K; \$9.95.

What is a **REVERSIBLE GAME**? You can stop the play at any point, back up and then do an "instant replay", analyzing your strategy. Or back up and resume the game at an earlier point, trying out a different strategy. Reversibility makes learning a challenging new game more fun. And helps you become a skilled player sooner.

WORLD OF ODYSSEY—a new adventure game utilizing the full power of Disk II, which enables the player to explore 353 rooms on 6 different levels full of dragons, dwarfs, orcs, goblins, gold and jewels. Applesoft II 48K; \$19.95 includes diskette.

PERQUACKEY—an exciting vocabulary game which pits the player against the clock. The object of the game is to form words from a group of 10 letters which the computer chooses at random. The words must be 3 to 10 characters in length with no more than 5 words of any particular length. Each player has only 3 minutes per turn. The larger the words the higher the score. Applesoft II 16K; \$9.95.

APPLESHIP—is a naval game in which two players enter their ships in respective oceans. Players take turns trying to blast their opponent's ships out of the water. The first player to destroy their opponent's ships may win the game. A great low-res graphics game. Applesoft II 32K; \$14.95.

Available at your
local computer store

Call or write for our free
SOFTWARE CATALOG

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

DEALER INQUIRIES INVITED

POWERSOFT, INC.

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

Programs Available on Diskette
at \$5.00 Additional

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

Tape Execute File Create and Use

Once upon a time, a computerist wanted to convert his integer BASIC programs to Applesoft BASIC. He read about a great technique - but it required a disk. He did not have a disk, but did have cassettes. Could the technique be modified for tape? And, what other changes would be required for the complete conversion? Some interesting things were discovered, and are reported here.

Allen J. Lacy
1921 Oglethorpe Avenue
Albany, GA 31707

For a long time, I had been trying to find a way to convert Integer programs to Applesoft. So it was with great interest that I read the How to Section titled "Disk Magic;" in Contact 5. A short summary follows for those who didn't get Contact 5. It was a way to list the Integer programs on to disk and then load it into Applesoft. This was done by placing the following line in the program:

```
0 PRINT"@ OPEN X":POKE 33,33:  
PRINT"@ WRITE  
X":LIST:PRINT"@CLOSE X":END
```

(Where @ means Control D)

When this line is entered type "RUN" and press "RETURN". When the operation is complete, enter Applesoft and EXECute the file.

The only problem with this method

is that I do not have a disk yet.

I started to think about how this could be done with just a tape. During a normal "SAVE" both Integer and Applesoft write the program to tape the way it is stored in memory, not the way it is listed. The program is stored as tokens; and since the tokens do not match, Applesoft cannot load Integer programs.

So I wrote two routines which link into the input and output hooks CSW and KSW at \$36-\$39. (This article uses "\$" to indicate a hexadecimal number.)

The output routine gets each byte as the Apple outputs it and stores it in a buffer before the actual output. When the Apple outputs a carriage return, the routine writes the buffer to tape. This continues until the Apple outputs a car-

riage return as the first character, the routine then resets the output hook.

The buffer is 256 bytes long. This number was chosen because that is the length of the Apple's input buffer. Note the buffer is from \$3F00 to \$3FFF (decimal 16128 to 16383). This is because my Apple has 16K. For different memory sizes this can be changed. If you have an assembler, change the SAVE address to the values in table 1. If you do not have an assembler, change the locations shown in table 2.

The input routine reads the tape records back into memory and passes the bytes through the input hooks. This continues until a record comes in which contains a carriage return as the first byte, the routine then gives control back to the keyboard.

MEMORY SIZE	SAVE ADDRESS	HIMEM
32K	\$7F00	32512
48K	\$BF00	—16640

Applesoft
A = 16 * RND(1)

In Integer either “#” or “<” can be used to mean not equal, in Applesoft only “>” can be used.

Example:

Integer
IF A # B THEN 10

Applesoft
IF A <> B THEN 10

There is no “MOD” operation in Applesoft, so you have to calculate the modulus.

Example:

Integer
B = A MOD C

Applesoft
B = A - INT(A/C) * C

Variable names may have to be changed. In Integer all letters are significant; in Applesoft only the first 2 letters are significant. To Integer PAY1 and PAY2 are different; to Applesoft they are the same variable.

Example:

Integer
PAY1 = PAY2 + PAY3

Applesoft
P1 = P2 + P3

Another difference is the way strings are handled. In Integer “DIM A\$(20)” means set up 1 string which can be up to 20 characters long. To Applesoft, it means set up 20 strings each of which can be up to 255 characters long. So all string dims should be removed from the program.

Also to get specific characters out of a string, you have to use the MID\$ function in Applesoft.

Example:

Integer
B\$ = A\$(2,5)

Applesoft
B\$ = MID\$(A\$,2,3)

The last difference that I have found is that all variables should be converted to Applesoft integer variables. This is not always needed, a lot of programs will run without this being done.

Example:

Integer Applesoft
A = B A% = B%

Table I

The first version used the tape write routine at \$FECD, which writes a 10 second header; therefore, the write took about 11 seconds, 10 for the header and 1 for the data. However, I noticed that at \$FECD the instruction is LDA #\$40 followed by JSR HEADR. Therefore when I want to write a record to tape, I load the accumulator with \$20 and enter the monitor at \$FECD. This causes the Apple to write a 5 second header, which means each record takes 6 seconds.

To use:

Load the routines into memory
Enter Integer Basic
Type “HIMEM:16128”
Press “RETURN”
Load the Program
Type in the following line:
0 POKE 33,33:CALL 769:LIST:END
Type “RUN”
Set the recorder in record mode
Press “RETURN”

The program will now list to tape and the TV. When this has finished, the prompt (P) will reappear.

Now enter Applesoft
Rewind the Tape

Warning: Since the headers are only 5 seconds long, you must set the tape as close to the beginning of the first one as you can.

Type “HIMEM:16128”
Press “RETURN”
Type “CALL 772”
Start the recorder in play mode
Press “RETURN”

The program will come into Applesoft as if you had typed it in. When the Applesoft prompt (P) appears with just the cursor behind it, control is back at the keyboard.

Now what you have to do is change the things which are different between Applesoft and Integer. This will have to be done whether you use the disk or tape.

All “TAB” statements have to be changed to ‘HTAB’
All computed “GOTO” and ‘GOSUB’ have to be changed to “ON” “GOTO” or “ON” “GOSUB”.

Example:

Where N can vary from 1 to 4
Integer
GOTO 400 + N * 100

Applesoft
ON N GOTO 500,600,700,800

All multi statement “IF”s will have to be broken into two lines because of difference in the way Integer and Applesoft handle ifs.

Example:

100 IF A = B THEN A = A + 1:C = C + 1

In Integer C always has one added to it, whether or not A equals B. This same line in Applesoft will cause C to have 1 added to it only if A equals B. So for the program to work like the Integer program, the line will have to be broken into two lines.

100 IF A = B THEN A = A + 1
101 C = C + 1

The random number functions are different between Integer and Applesoft.

Example:

Integer
A = RND(16)

MEMORY SIZE	\$30C	\$322	\$34D	\$369	HIMEM
32K	\$7F	\$7E	\$7F	\$7E	32512
48K	\$BF	\$BE	\$BF	\$BE	—16640

Table II


```

1000 *****
1010 *
1020 *           TAPE EXECUTE FILE *
1030 *           CREATE & USE *
1040 *
1050 * MAIN USE TO CONVERT INTEGER *
1060 * PROGRAMS TO APPLESOFT II *
1070 *
1080 *           BY *
1090 *           ALLEN J LACY *
1100 *           AUGUST 1979 *
1110 *
1120 *****
1130 *****
1140 *
1150 * 256 BYTE BUFFER TO STORE TEXT *
1160 * FROM ADDRESS $3F00 TO $3FFF *
1170 * CHANGE FOR LARGER APPLE II *
1180 *
1190 *****
1200 SAVE .EQ $3F00
1210 PT .EQ $300
1220 KSW .EQ $FD1B
1230 WR .EQ $FECE
1240 RE .EQ $FEFD
1250 COUT .EQ $FDD0
1260 XSAV .EQ $47
1270 .OR $301
1280 *
1290 *****
1300 *
1310 * SET UP FOR OUTPUT OF FILE *
1320 *
1330 *****
1340 STP JMP SP
1350 *
1360 *****
1370 *
1380 * SET UP FOR INPUT OF FILE *
1390 *
1400 *****
1400 STK JMP SK
1420 *
1430 *****
1440 *
1450 *SUBROUTINE TO SET TAPE POINTERS*
1460 *
1470 *****
0307- A9 00 1480 SET LDA #SAVE
0309- 85 3C 1490 STA $3C
030B- A9 3F 1500 LDA /SAVE
030D- 85 3D 1510 STA $3D
030F- A9 FF 1520 LDA #SAVE+255
0311- 85 3E 1530 STA $3E
0313- A9 3F 1540 LDA /SAVE+255
0315- 85 3F 1550 STA $3F
0317- 60 1560 RTS
1570 *
1580 *****
1590 *
1600 * BYTE OUTPUT ROUTINE *
1610 *
1620 *****
0318- 86 47 1630 PRT STX XSAV STORE X REG
031A- FE 00 03 1640 INC PT
031D- AE 00 03 1650 LDX PT
0320- 9D FF 3E 1660 STA SAVE-1,X STORE BYTE
0323- C9 8D 1670 CMP #S8D CP?
0325- F0 06 1680 BEQ CR
0327- A6 47 1690 LDX XSAV RESTORE X REG
0329- 20 F0 FD 1700 JSR COUT
032C- 60 1710 RTS
032D- AD 00 03 1720 CR LDA PT
0330- C9 01 1730 CMP #1
0332- F0 15 1740 BEQ NPT
0334- 20 07 03 1750 JSR SET
0337- A9 20 1760 LDA #S20 5 SEC HEADER
0339- 20 CF FE 1770 JSP WR WRITE TO TAPE
033C- A6 47 1780 LDX XSAV RESTORE XREG
033E- A9 8D 1790 LDA #S8D OUTPUT CR
0340- 20 F0 FD 1800 JSR COUT
0343- A9 00 1810 LDA #0 RESET PT
0345- 8D 00 03 1820 STA PT
0348- 60 1830 RTS
0349- A9 8D 1840 NPT LDA #S8D
034B- 8D 00 3F 1850 STA SAVE
034E- 20 07 03 1860 JSR SET
0351- A9 20 1870 LDA #S20
0353- 20 CF FE 1880 JSR WR WRITE LAST REC
0356- A9 F0 1890 LDA #COUT RESET PRINT

```

```

0358- 85 36 1900 STA $36
035A- A9 FD 1910 LDA /COUT
035C- 89 37 1920 STA $37
035E- 60 1930 RTS
1940 *
1950 *****
1960 *
1970 * BYTE INPUT ROUTINE *
1980 *
1990 *****
035F- 86 47 2000 RED STX XSAV SAVE X REG
0361- EE 00 03 2010 INC PT
0364- AE 00 03 2020 LDX PT
0367- 9D FF 3E 2030 LDA SAVE-1,X GET BYTE
036A- C9 8D 2040 CMP #S8D CR?
036C- D0 11 2050 BNE NCR
036E- AD 00 03 2060 LDA PT
0371- C9 01 2070 CMP #1
0373- F0 0D 2080 BEQ NKEY
0375- 20 AC 03 2090 JSR TR
0378- A9 00 2100 LDA #0
037A- 8D 00 03 2110 STA PT
037D- A9 8D 2120 RCR LDA #S8D LOAD CR
037F- A6 47 2130 NCR LDX XSAV RESTORE X REG
0381- 60 2140 RTS
0382- A9 1B 2150 NKEY LDA #KSW GIVE CONTROL
0384- 85 38 2160 STA $38 TO KEYBOARD
0386- A9 FD 2170 LDA /KSW
0388- 85 39 2180 STA $39
038A- 4C 7D 03 2190 JMP RCR
038D- A9 5F 2200 SK LDA #RED GIVE INPUT
038F- 85 38 2210 STA $38 CONTRCL TO
0391- A9 03 2220 LDA /RED RED
0393- 85 39 2230 STA $39
0395- 20 AC 03 2240 JSR TR READ 1ST REC
0398- A9 00 2250 LDA #0
039A- 8D 00 03 2260 STA PT
039D- 60 2270 RTS
039E- A9 18 2280 SP LDA #PRT GIVE OUTPUT
03A0- 85 36 2290 STA $36 CONTROL TO
03A2- A9 03 2300 LDA /PRT PRT
03A4- 85 37 2310 STA $37
03A6- A9 00 2320 LDA #0
03A8- 8D 00 03 2330 STA PT
03AB- 60 2340 RTS
03AC- 20 07 03 2350 TR JSR SET
03AF- 20 FD FE 236C JSP RE
03B2- 60 2370 RTS
2380 .EN

```

The Apple Shoppe

JOURNAL OF APPLE APPLICATIONS Vol. No.

EDITED BY
DAVID E. SMITH

PUBLISHED BY **COMPLUTUTOR**

YOU BOUGHT THE BEST! NOW LEARN TO USE IT!

AT LAST!

A magazine devoted to Applications as well as Technique for the Apple Computer.

THE APPLE SHOPPE WILL TEACH YOU HOW TO DO ALL THOSE FANCY THINGS ON THE APPLE. LEARN HOW OTHERS ARE USING THEIR APPLES IN THE HOME, SCHOOLS AND BUSINESSES.



CHECK THESE FEATURES:

- ✓ Feature Articles on Apple Applications
- ✓ Program of the Month — How To with Listings
- ✓ New Products Review — All Based, Priced, and Language Lab — Learn Basic, Pascal, Fortran, and more!
- ✓ Future Projects — Participate in a new group project called "The China Syndrome"
- ✓ Graphics Workshop — Local projects from the new group to Super Programs

YES I want to learn how to get the most out of my Apple. Send me a one year subscription. I enclose \$12.

NAME: _____

ADDRESS: _____

CITY _____ STATE _____ ZIP _____ PHONE _____

NO, I already know it all, but send me a free sample of next issue.

Send check or money order to: Apple Shoppe, P.O. Box 701, Placentia, CA 92670 or call (714) 998-0441



compas
microsystems

P.O. Box 687
224 S.E. 16th Street
Ames, Iowa 50010

DAIM



DAIM is a complete disk operating system for the ROCKWELL INTERNATIONAL AIM 65. The DAIM system includes a controller board (with 3.3K operating system in EPROM) which plugs into the ROCKWELL expansion motherboard, packaged power supply capable of driving two 5 1/4 inch floppy drives and one or two disk drives mounted in a unique, smoked plastic enclosure. DAIM is completely compatible in both disk format and operating system functions with the SYSTEM 65. Commands are provided to load/save source and object files, initialize a disk, list a file, list a disk directory, rename files, delete and recover files and compress a disk to recover unused space. Everything is complete — plug it in and you're ready to go! DAIM provides the ideal way to turn your AIM 65 into a complete 6500 development system. Also pictured are CSB 20 (EPROM/RAM) and CSB 10 (EPROM programmer) which may be used in conjunction with the DAIM to provide enhanced functional capability. Base price of \$850 includes controller board with all software in EPROM, power supply and one disk drive. Now you know why we say —

There is nothing like a

DAIM

Phone 515-232-8187

Why a PET, APPLE, 6502 BASIC Compiler? A Simple Explanation

BASIC, on almost all 6502 microcomputers, is run with an Interpreter. A more efficient method of running BASIC is through a Compiler. This article discusses what a Compiler is, how it works, and discusses a BASIC Compiler currently under development.

Bruce M. Beach
Horning's Mills
Ontario, LON 1JO
Canada

A group of Canadian PET users are developing a compiler for the PET that will also be usable on the APPLE or any 6502 based computer. This may be a very significant step in regards to the usefulness of the PET.

This article answers the questions indicated in its sub-headings. So as not to waste your, the reader's, time, you should just go to those sub-headings to which you do not know the answer.

The Topics Being Covered are:

1. What is a compiler?
2. What is the difference between a compiler and an interpreter?
3. What is the difference between a direct compiler and a p-code compiler?
4. Why would a BASIC compiler be so useful on a PET?
5. What is the status of the CANPET BASIC compiler?

What is a Compiler?

A compiler is a computer program which takes a set of instructions, written according to some set of rules, and transforms it into a machine language computer program, a string of binary characters. This is the *real machine language*. Everything actually stored in the machine can be represented by a combination of 1 and 0 digits.

Early computers built in the 1950's were programmed with strings of binary numbers and it was extremely difficult to tell where an error had been made in a long binary string as, 10111010110101. There are convenient methods of converting binary numbers to other number bases such as octal, hexadecimal, or decimal. Thus programmers were able to use more recognizable numeric strings such as, 73 (Octal) or A2 (Hex) to represent their Instruction Code. Operations performed by a computer (such as add, subtract, or move data from one location to another) have specific operation codes assigned to them. Some computers have as many as four hundred different operations (op-codes) in their instruction set.

Because it was still easy for a programmer to become confused about what the numbers represent, a still more simplified method of representing programs was developed using what are called mnemonics (nuhh-monics). For example, the letters AD might be used for add, SB for subtract, and LDA for load register A. This method of writing programs is sometimes mistakenly called machine language programming; in fact, together with symbolic addressing, it is Assembler Language Programming.

A program has to be available that will recognize the mnemonics of the assembly language instructions, translate them into the appropriate op-codes, and allocate actual storage locations for those represented by the pro-

grammer as symbolic names. Such a program is called *an assembler*. If such a program (an assembler) is not available and the operating instructions are written using only numeric code, the program is said to have been "hand assembled".

More powerful assemblers keep track of address locations in programs and may provide various helpful debugging aids. However, even the most powerful assemblers still require an understanding of assembly language in order to use them; and more importantly still, the more powerful they are the more likely they are to be untransportable. That is to say they are unlikely to be able to move from one model of a machine to another because they usually gain their "macropower" from features inherent in a particular machine.

Because a great deal of skill and effort is required to write a program in assembly language, new languages called higher level languages were designed to make life easier. The first widely used such higher level language was FORTRAN (FOR-mula TRAN-slater) used mainly by the mathematically oriented. The FORTRAN compiler allowed the programmer to express his problem in rather conventional looking mathematical notation and then took the program *SEE BOX* and converted it into assembly language instructions or directly into Machine Code.

Another high level language, COBOL (Common Business Oriented Language), was developed for accountants and the business community which allowed these professionals to express their computer programs in expressions easily learned by them. The COBOL compiler (a program written in machine language) took the user's program written in COBOL and compiled it into an executable machine language program. Other well known languages which require compilation are "C", FORTH and PASCAL. Compilers have been or are being developed for the PET for the languages "C", FORTH and PASCAL, but to date there has been no compiler for the full BASIC language. The following discussion will point out the usefulness of such a compiler and tell you when and where one will be available.

What is the Difference Between a Compiler and an Interpreter?

The code which a programmer writes in a higher level language is called the source code and the output from the compiler, which processes that code, is called the object code. In the process of making the conversion a compiler may have to make several "passes", i.e., complete scans through the source code, so compilers are often distinguished as being single or multiple pass compilers. It usually takes a multiple pass compiler longer to compile than a single pass compiler but the multiple pass compiler might be preferable if, for example, the object code it generates is more efficient.

In any case, once the compiler has completed its task the object code can be saved and used over and over again without recompiling. Interpreters, such as the BASIC Interpreter found in the PET and other popular micro-computers, do not work in this manner. They take the user's source program, written in the higher level language (the BASIC statements), and analyze (interpret) each statement one at a time to determine its equivalent machine code, and then execute this code. Moreover, *and this is the chief drawback to interpreters*, they do not save the object code. The next time that BASIC statement is executed the machine again has to interpret that line of code. For example, if there is a FOR...NEXT loop in the program that contains six statements between the FOR and the NEXT and the loop is to be executed 100 times, then each of those six lines of code will be interpreted (translated) into machine language 100 times. This results in a total of 600 translations made by the interpreter, whereas the compiler would have made only six. In both cases, the machine code is performed 600 times; but in the interpretation, the analysis represents a

significant overhead which is absent in the compiled version.

Purists may object that this is a somewhat simplified explanation because, in fact, the interpreter stores token (numbers) for the BASIC keywords, and often jumps to predefined specific runtime routines rather than assembling new code. However, in principle the interpreter works in this manner and for this reason interpreted programs are 10 to 100 times slower in execution than compiled programs. Another factor which often slows down an interpreter is that it must repeatedly do much error checking that a compiler does only one time.

The advantage of an interpreter, however, is that one need not wait for the compile to take place before execution. So long as high speed in program execution itself is not needed, an interpreted program may perform quite fast enough; and although there may be other reasons (some of which will be described later) that may make compilation desirable, it is apparent that an interpreter will reduce the time required for program development.

There are advantages to an interpreter besides convenience in programming. Source code requires much less memory than object code. A single concise BASIC statement such as:

$$\text{If } X = (Y * L) / M \text{ THEN } R = X + (M - L),$$

expands through compilation into many machine language statements. Consequently a much longer program can be written in BASIC and stored in a small computer that interprets each line and "throws away" the object code immediately after it is used, than in a machine that has to store all the object code before execution begins.

Incidentally, there are many high level languages that are not general purpose programming languages. RPG's (Report Program Generators), for example, are high level languages used to format reports. There are also many DBM (Data Base Management) languages (such as ADABAS, MARK IV, etc.) that are used to access large files of data. On the surface these programs appear very similar to the languages processed by compilers as regards the syntactical rules they require for input. That is to say the user writes a "program" for his application that is in many ways like a computer program that he would write for a compiler. However, while these systems do what they are designed to do very well (i.e., access some particular data base), they are not general purpose languages and cannot be used efficiently for many purposes that a compiled language can.

To summarize then, a *compiler translates the source code* into object code one time which is then used over and over again; whereas an interpreter, such as PET's BASIC, "throws away" the object code after each execution and then must re-translate again from the source before an instruction can be used again. The advantage of using an interpreter is that it takes much less memory to store a whole program in source format rather than in object format, and execution takes place immediately rather than waiting for a complete new re-compile after each program change. However, among its other advantages, a compiled program can be executed at more than ten times the speed of an interpreter and this is often critical in certain applications.

What is the Difference Between a Direct Compiler and a P-Code Compiler?

A compiler then, takes the source language code of a particular high level language and translates it into object code—that is to say, into the machine language op-codes. Because a computer always automatically executes the next instruction following the one it is presently executing (unless there is a branch), it is much faster not to have any branches. However, code written without branches would usually require more memory than is available internally to the computer. Also, it would not take advantage of the "conditional" branching or decision making power of the computer which is the essence of a program.

Consequently, *one of the major design decisions* in designing a compiler is the trade-off between using memory-consuming repeating code "inline" to save branches and increase speed, or making time-consuming repeated branching to the same sub-routines in order to conserve memory. A JSR (Jump-to-Sub-Routine) requires the computer to save from the PC (Program Counter) the next address it would have executed in sequence, and load instead in the program counter the address of the sub-routine instruction. On RTS (Return from Sub-routine) the instruction address that was originally saved must then be restored to the PC. If there were only a few instructions in the sub-routine, there will be no saving of memory and time will be wasted in going to the sub-routine. The computer instead simply could have processed the next couple of instructions. However, if the sub-routine contains many instructions, memory will be saved by going there at the expense of a little time for making the branches. It all depends on the relative value of speed and memory in a particular system.

A compiler designer soon finds that

certain large blocks of code are used repeatedly. Therefore, every time a source program requests a certain type of activity the compiler causes the object code to jump to the specified block of code that can handle that activity. Sometimes two activities are similar, although not identical; but if the code for each is very long and the differences are minor, it is frequently more efficient to *generalize* the code and then to distinguish between the differences of the activities within the block of code. Now again, there are some trade-offs most likely requiring some additional branches for each of the activities that would not be necessary if they had their own unique code. We are in fact "interpreting" at execution time some of the code within the compiler-generated code. This then is not true object code for what was the source statement but is in a very limited sense Pseudo Code (P-Code).

While this type of approach is present to some extent in almost all compilers, some compilers make heavy use of this approach. The generalized code that will interpret the specific statements generated as object code by the compiler amounts to an "overhead" in both usage of memory and in execution time.

Some FORTH and PLM compilers currently available for the PET are so heavily dependent on these techniques that the resulting object code executes as little as 3 times as fast as the BASIC Interpreter. These same compilers require several K overhead in memory for the specialized routines that consequently become a part of all programs, whether they are actually used or not. This can be very detrimental in some important situations.

It is possible to write a compiler that is resident in memory and interprets all of the code at execution time. In such a case we have come full circle and have what we started with—an interpreter. This is indeed why many of the so-called compilers perform little better than an interpreter.

How, then, can one tell whether or not they have a "true" direct compiler or a largely P-code simulator? The answer is by *benchmarking*. Because there are different design philosophies behind different compilers, one must take a compiler and compare it to the other alternatives (i.e., other compilers or the interpreter). One does this by writing a test program with statements similar to the type they use in actual applications. Perhaps for one user there are lots of loops and string handling. Another user may particularly use math functions and arrays. The particular test program is then run using both products and the results are compared. Only in this way

will you know which of the two products will perform better in terms of compile-time and/or execution-time. Other important considerations may be maintenance, direct access to the object code to allow modification, types of statements available, ease of operation, documentation, support, expected improvement or obsolescence, etc.

To summarize this section then, a "direct-compiler" uses relatively less pseudo-code and executes faster than straight P-code compilers. Performance can only be determined by benchmarking for specific applications.

Why Would a Basic Compiler be So Useful on a PET?

Aside from the considerable speed improvement that can be obtained from a well compiled program, are there any other advantages to using a BASIC compiler? Most definitely, yes. However, before elaborating let us pursue the question of speed itself. For many applications the PET's BASIC interpreter's speed is entirely adequate. It is in real time applications (such as process control, where the PET is monitoring some other device attached to it through an interface such as the IEEE-488 Port, located on the back of the machine) that greater speed is needed. Since only the PET among popular personal micro-computers has the necessary IEEE Port for attaching many laboratory and technical devices, faster programs are also more significant to PET users.

There are a number of S-100 bus applications that could benefit from increased program speed and these can be implemented on the PET through available S-100 interface boards. Often only a few specific routines need the higher speed afforded by assembly language programming and this could be accomplished by writing those few routines in assembly language and doing a SYS call from the BASIC program to them. This latter approach, however, still requires that the programmer understand assembler language; whereas, by using a BASIC compiler, he needs only know BASIC.

Where a BASIC compiler can really shine is in the development of marketable systems. There are now available for less than \$300, systems that interface directly with the PET that can be used for burning PROMS (programmable read only memories) and E-PROMS (ultra violet erasable proms). These chips will fit into socket holders inside the new 16K PETS, or the socket holders in the expansion boards on the old 8K PETS, and can hold a machine language program in readiness for a user, even when the computer is turned off—they are called non-volatile.

This makes it convenient for users still to be able to use their computer in just the same way as any other PET user. But at the same time, they are able to step up to a PET that contains a PROM programmed for a specific task and immediately access the special program without having to wait for it to load from a tape or disk. In addition to making the computer easier for the user to use, this is also a very convenient way for the developer to distribute his program and makes copying of it much more difficult than if it were on tape. The producer's incremental cost of duplicating programs for distribution using PROMS should be well under \$20 each.

More importantly, we can go one step further and take a program that has been written on the PET in BASIC, compiled, and stored in PROM and use that PROM along with a 6502 micro-processor to build up an entirely separate device that no longer involves the PET at all. In this way the PET has become a *very powerful* development tool for the garage or basement inventor that is comparable to similar development systems that have been used in industry for the last several years but that have cost *many* thousands of dollars. A true BASIC direct compiler will therefore allow serious PET users to develop from PROMS, faster and more memory efficient object code *while using the power of the present PET BASIC interpreter* for rapid program development.

What is the Status of the CANPET BASIC Compiler?

In June 1979 work was begun on a PET BASIC direct one-pass compiler. The language supported by this compiler is intended to be identical with that supported by the PET BASIC interpreter with the exception of dynamic array declaration/allocation.

The Co-ordinators of the project, Mr. Bruce Beach and Mr. Brian Beswick, have retained the service of a Toronto-based consulting firm with nearly 15 years of software experience and expertise in compiler design. Assistance is also being given by interested and knowledgeable individuals in the Canadian PET community, such as Mr. Jim Butterfield.

The first pre-releases of the compiler should be available for use by the time this article appears in print. Initial users will be sought in a wide diversity of applications so that the compiler's performance can be critically evaluated. Any persons who feel they would like to participate in the early evaluation process are invited to contact the Author.

Article Summary

A BASIC direct compiler that makes minimal use of P-code is being developed for the PET and APPLE or any 6502 based computer by a private Canadian group. It is anticipated that the resulting object code will require more storage than the source BASIC provided to the interpreter but less than that generated by other presently available compilers.

The chief advantage of the new compiler is that its resulting code should execute many times faster than the speed obtained by using the PET or APPLE's BASIC interpreter.

The new compiler in combination with the present powerful PET and APPLE BASIC interpreters should greatly facilitate the development of new systems that take advantage of the PET's and APPLE's 6502 microprocessor and the PET's IEEE-488 Port compatibilities.

Serious users who would be willing to help benchmark and critically evaluate the performance of this new

BASIC compiler are invited to contact the author, Mr. Bruce M. Beach, Horning's Mills, Ontario L0N 1J0,

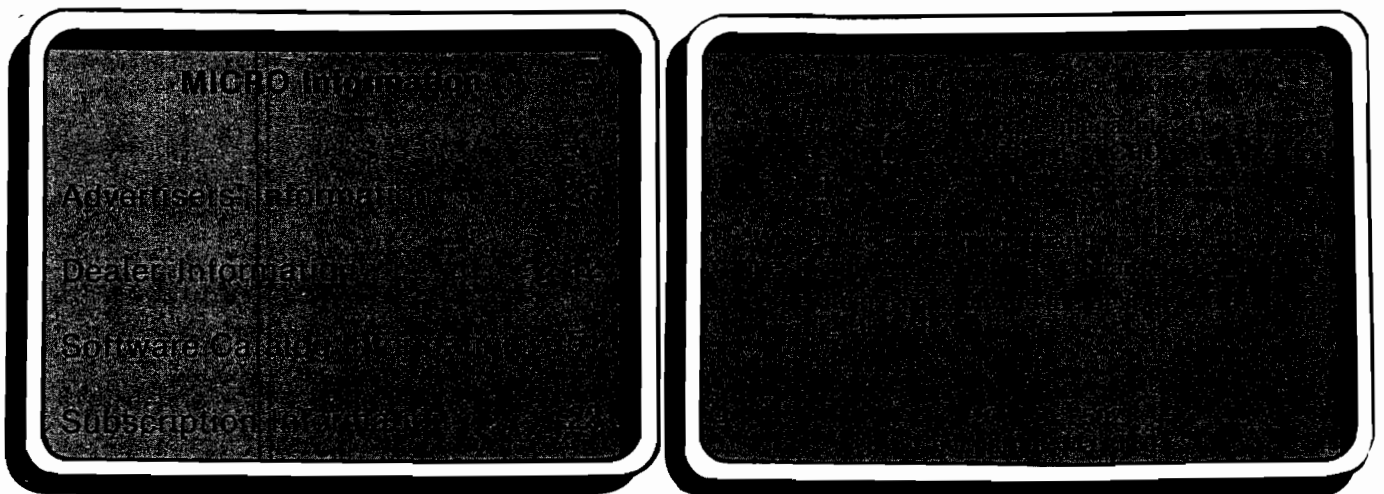
Canada, (519)925-6035, or Mr. J. Brian Beswick, 1755 Rathburn Road, Unit 45, Mississauga, Ontario L4W 2M8 (416)624-5225.

```
DO 300 I=1,N
IF (CR(I).EQ.0) GOTO 300
IF (STRTSW(I).EQ.0) GOTO 407
CR(I)=0
C CHECK IF INFORMATION COMING
IF (STEP(I).GT.6) GOTO 301
BFPTR(I)=BFPTR(I)-1
C YES INFOIRMATION
T=STEP(I)
DO 300 J=1.BFPTR(I)
INFO(I,T,J)=BUFFER(I,J)
300 CONTINUE
STOP
END
```

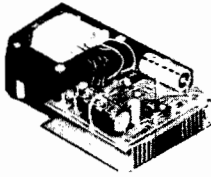
Figure 1: Example of Fortran Routine

```
IF FEMALE GO TO WOMAN
ELSE GO TO MAN.
WOMAN. IF WEIGHT < MIN-FEMALE-WT (J)
SUBTRACT WEIGHT FROM MIN-FEMALE-WT (J) GIVING LBS-U (NU)
GO TO SKINNY.
IF WEIGHT > MAX-FEMALE-WT (J) SUBTRACT MAX-FEMALE-WT (J)
FROM WEIGHT GIVING LBS-OV (NOV)
GO TO FAT.
GO TO NORMAL.
MAN. IF WEIGHT < MIN-MALE-WT (J)
```

Figure 2: Example of Cobol Routine



POWER A PLUS™

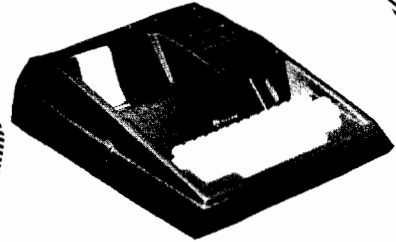


Designed specifically for the AIM-65
 +5V @ 5A and +24V @ 1A
 110V/60Hz or 220V/50Hz
 Short Circuit protection
 Over voltage protection
 Enough power to run fully loaded AIM
 plus several additional boards **\$55⁰⁰**

AIM ENCLOSURE™

AIM ENCLOSURE:
 A neat, safe package for the AIM 65
 Room for one expansion board:
 Memory Plus, Video Plus,
 or Proto Plus
 Enclosure alone: **\$45⁰⁰**

AIM PLUS™



With Built-in POWER SUPPLY:
 Power A Plus supply
 On/off switch with pilot light
 Air vents in enclosure to allow heat
 dissipation
 Fully Assembled, Ready to Go **\$110⁰⁰**

AIM ALL YOUR NEEDS AT US!!

MEMORY PLUS™

Modified for AIM to address at 1000
 (4K) boundary - at no charge.
 8K Static RAM
 Sockets for 8K EPROM
 EPROM Programmer
 6522 I/O Port
 Over 1200 in use. **\$200⁰⁰**

AIM 65

VIDEO PLUS™

Glass Teletype Software for AIM is
 available now and it's FREE. It works
 with the AIM Monitor and BASIC
 without any user programming.
 UPPER/lower case ASCII
 2K Display RAM expandable to 4K
 Provision for 2K EPROM
 Add up to 128 Programmable
 Characters by plugging in RAM
 Supports ASCII Keyboard
 Fully Assembled and Tested: **\$245⁰⁰**

MOTHER PLUS™

The Sensible Way to Expand:
 Fully Buffered and Decoded
 Standard Bus Structure
 Add up to five expansion boards
 Audio/TTY connectors, Port
 Socket
 Fully Assembled and Tested: **\$80⁰⁰**

CAGE PLUS™



Use with the AIM 65 and MOTHER
 PLUS
 Aluminum frame with card guides
 Holds five additional boards with
 your AIM on top
 Package your expanded AIM 65.
\$20⁰⁰ with Mother Plus, \$25⁰⁰ alone.

PROTO PLUS™

Add Special Circuits to your system.
 Provisions for:
 40 14/16 pin sockets
 4 24/40 pin sockets
 3 voltage regulators
 Misc. other components
 Two sets of gold plated dual 22 finger
 connectors with same spacing as AIM.
 Wire wrap or solder connections.
 Size: 8 1/4 x 10 3/4 **40⁰⁰**

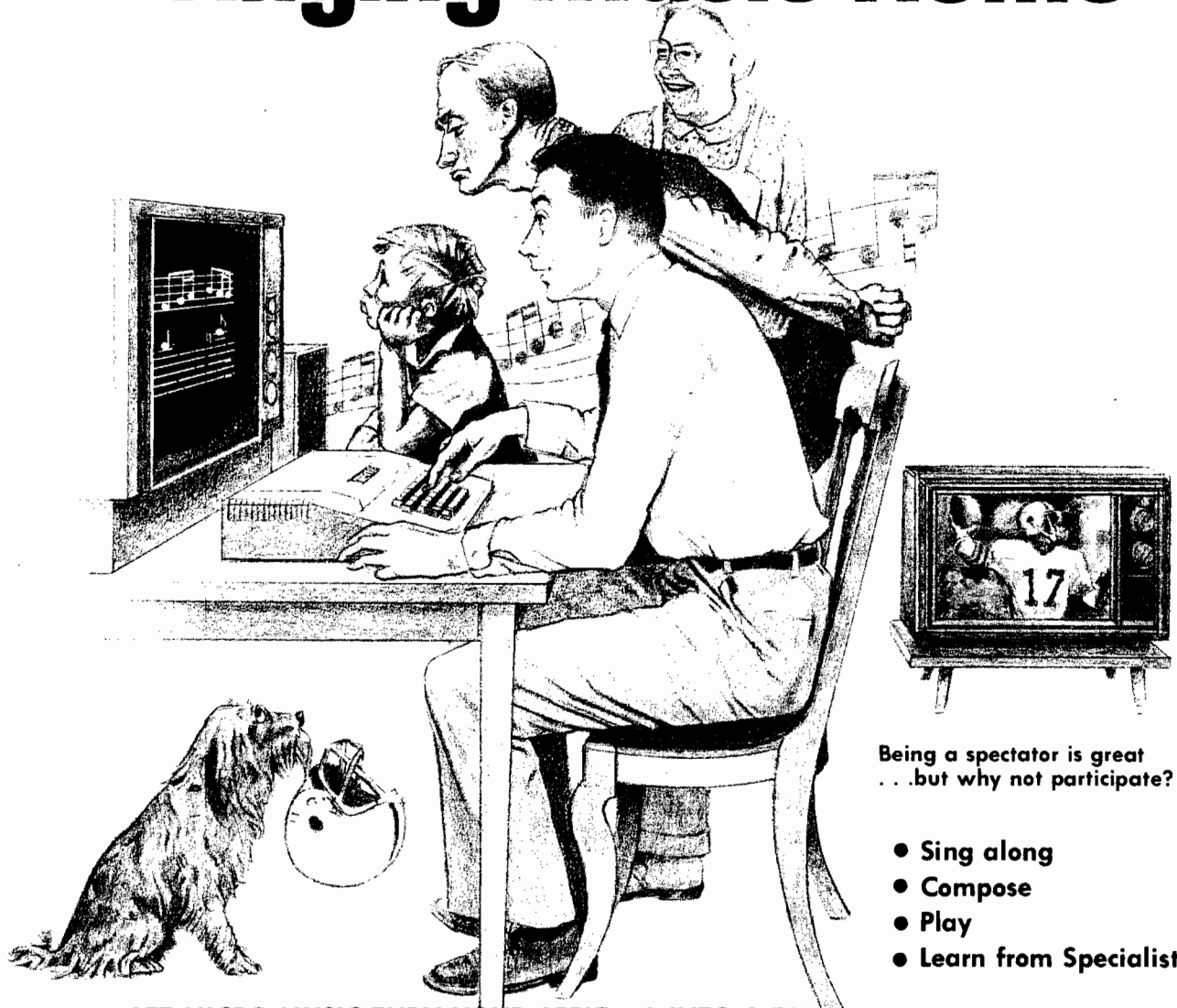
Documentation available for MEMORY
 PLUS (\$10⁰⁰), VIDEO PLUS (\$10⁰⁰) and
 MOTHER PLUS (\$5⁰⁰). Documentation
 cost will be applied to purchase price.
 Prices quoted do not include shipping
 and handling charges. Please call or
 write for complete catalog.

THE COMPUTERIST INC

P.O. Box 3
 So. Chelmsford, MA 01824
 617/256-3649

When ordering, please specify the
 MEMORY PLUS modification and the
 VIDEO PLUS Glass Teletype software
 for your AIM 65 at no charge.

Bringing Music Home



Being a spectator is great
...but why not participate?

- Sing along
- Compose
- Play
- Learn from Specialists

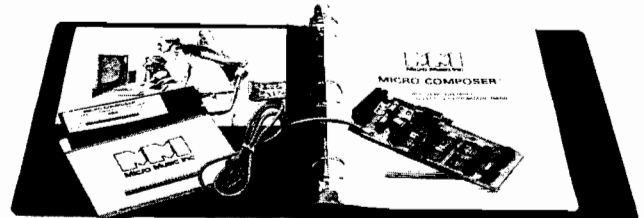
LET MICRO MUSIC TURN YOUR APPLE II® INTO A FAMILY MUSIC CENTER!

VISIT THE APPLE DEALER NEAREST YOU AND ASK FOR A DEMONSTRATION OF MMI'S MICRO COMPOSER™
The MICRO COMPOSER LETS YOU—

- Play up to 4 simultaneous voices
- See all 4 voices at the same time you're hearing the music—a must for music editing!
- Enter music notes by a fast, simple and well-tested coding system.
- Program the pitch, rhythm, and timbre of the music. Tempo is varied by the Apple paddle.
- Choose 7 different tone colors for each voice or create your own tone color.
- Compose, edit, display, and play music through an interactive, command-driven language that's easy to learn.
- Save your music on disk or cassette.
- Hear quality music sound at low cost through the MICRO MUSIC™ DAC card. No amplifier needed! Designed for MMI by Hal Chamberlin and Micro Technology Unlimited.
- Select from future MMI music instruction software to accompany the MICRO MUSIC DAC.

Ask your local dealer for information on MMI products, or contact:

The MICRO COMPOSER is an APPLE II® compatible, low-cost music system designed by the folks at MMI. Our music software was designed by leading experts in music education. A simple step-by-step instruction manual leads you through entering, displaying, editing, and playing music with up to four voices—soprano, alto, tenor, and bass. You can change the sound of each voice to reed, brass, string, or organ sounds and you can even color your own music sounds!



HAVE FUN! THE MICRO COMPOSER comes complete with an instruction manual, software disk or cassette—in either Integer or Applesoft ROM BASIC, and the MICRO MUSIC DAC music card. Just plug the MICRO MUSIC DAC into the APPLE extension slot and connect the audio cable to a speaker.

Suggested retail price \$220.



Micro Music Inc 309 Beaufort, University Plaza, Normal, IL 61761

APPLE II is a trademark of Apple Computer Inc.

Human Physiological Parameters

One of the most common complaints about the home computer is that it does not really do much for the average consumer. After you balance your checkbook, then what? Here is a program, based on scientific data and studies, which calculates the proper weight for an individual as a function of height, body build, and sex. Written in Applesoft BASIC, it should be easily adapted to any other reasonable BASIC.

Dr. L.S. Reich
3 Wessman Drive
West Orange, NJ 07052

Introduction

The focus of public interest in nutrition has changed markedly during the past decade. In the past, the emphasis was on eating more of everything. Increasingly, the message is to eat less. The reason for the turnabout is that many foods are believed to be factors in causing or promoting such degenerative diseases as heart disease, diabetes, etc. Diet is also involved in an especially prevalent disease, obesity (excessive weight).

Excessive weight is associated with cardiovascular and renal diseases, diabetes, degenerative arthritis, gout, etc. On the basis of life insurance statistics, the most nearly ideal weight to maintain throughout life is that which is proper at age 25 for one's height and body build. Thus, height-weight tables

no longer indicate figures beyond ages of 25-30 years. A deviation of not more than 10 percent above or below the desirable weight for a given individual is not considered significant. The term overweight is applied to persons who are 10-20 percent above desirable weight; obesity is applied to persons about 20 percent or more overweight. Underweight generally applies to those individuals who are more than 10 percent below the established standards. Those who are more than 20 percent below such standards are considered to be seriously underweight.

Height-weight tables provide only approximations on the degree of fatness. More accurate measures of body fatness include measurements of thickness of subcutaneous tissue at designated body locations using calipers or by determination of body density by means of underwater

weighing. It has been estimated that about one-half of all men over 30 are at least 10 percent overweight and that one-quarter are obese. The incidence is higher for women, about 40 percent being obese by the age of 40.

Generally, the percent water in lean individuals is higher than in obese persons. The opposite is true in regard to body fat. The human body generally consists of 55-60 percent of body weight as water, about 17 percent as lipids (which includes fats), about 15 percent as protein, and about 1 percent as carbohydrates, about 5 percent of other materials. The total body water relative to body weight is usually lower in females than in males. Also, the predicted total body water has been found to be closely related to predicted surface area. Generally, the higher the weight-% of body water, the lower the weight-% of body fat.

PROGRAM LISTING

The Program

The program that follows indicates what a person should weigh based on height, body build and sex. The ideal weights given are generally for men and women of ages 25 and over. Besides ideal weights, this program estimates whether a person is obese, the percent that a person's weight is above the maximum ideal weight, the weight-% of body fat and of body water, and the body surface area. These physiological parameters are applicable to those over the age of 16.

Obesity is estimated by a critical obesity index based upon Quetelet's index (QI). This critical index is reached when the individual's weight is about 18 percent above the maximum ideal weight. Also, QI is used to estimate body fat (BF). The BF via QI is in good agreement with the value from weight-% water (BW) using the expression: $100-137 \cdot BW$ (however, another expression for BF is used in this program).

In the program listing that follows, REM statements are to be found in line numbers 20, 96, 100, 132, 138, 143, 148, 162, 200, and 490. Line numbers 500-600 contain height-weight data for females only while numbers 750-860 contain height-weight data for males only. In line number 50, W\$(J,K) denotes an array for heights and weights corresponding to small, medium, and large body frames. Line numbers 97-99 and 137 express the program limitations for females (must not have height below 5'0" or above 5'10", and if body frame is small, physiological parameters will not be given); while, lines 197-199 and 237 express the program limitations for males (must not have heights below 5'4" or above 6'3", and if body frame is small, physiological parameters will not be given). Line numbers 133 and 233 determine the percent that an individual's weight exceeds the maximum ideal weight and, numbers 145, 150-160, 245 and 250-260 calculate QI which is used to determine body fat and whether or not a person is obese.

Line numbers 165, 170, 175, 265, 270, 275 allow the estimation of body fat, body surface area, and body water both in men and women. Applesoft II BASIC in ROM was employed and the program required about 8.5K free bytes. (It may be noted here that a BASIC master command list has been published (*Recreational Computing*, Sept-Oct, 1979) which is applicable to SOL-20, PET 2001, APPLE II, and LEVEL II TRS-80 computers.)

```
2 HOME
3 PRINT "THIS PROGRAM TELLS YOU WHAT YOU SHOULD WEIGH BASED ON
  DATA ADAPTED FROM THE BOOK (WEIGHTS IN THIS BOOK WERE SUBTRACTED
  BY 3 TO GIVE WEIGHTS IN BED CLOTHING, WHICH WERE USED IN THIS
  PROGRAM), 'NORMAL & THERAPEUTIC NUTRITION' (13TH EDITION), ";
4 PRINT "BY C.H. ROBINSON, 1972, P.848 (MACMILLAN). HEIGHT
  LIMITATIONS ARE, FOR WOMEN: 5-0 TO 5-10; FOR MEN: 5-4 TO 6-3
  (NO SHOES). IDEAL WEIGHTS GIVEN ARE FOR BED CLOTHING AND ARE
  FOR ";
5 PRINT "MEN AND WOMEN OF AGES 25 AND OVER (FOR GIRLS 18-25,
  SUBTRACT 1 POUND FOR EACH YEAR UNDER 25).";
6 PRINT "BESIDES IDEAL WEIGHTS, THIS PROGRAM ESTIMATES OBESITY,
  BODY FAT, BODY SURFACE AREA, AND TOTAL BODY WATER. THESE ARE
  APPLICABLE TO THOSE OVER THE AGE OF 16. GENERALLY, THE % TOTAL
  BODY WATER IS LOWER IN FEMALES THAN IN MALES. ";
7 PRINT: PRINT: PRINT "PRESS 'CONT' TO CONTINUE!"; :PRINT: STOP:
  PRINT
8 PRINT "FURTHER, THE % OF WATER IN LEAN PERSONS IS HIGHER THAN IN
  OBESE PERSONS. ABOUT 55-60% OF THE BODY WEIGHT IS WATER. A
  DEVIATION OF NOT MORE THAN 10% ABOVE OR BELOW THE DESIRABLE
  WEIGHT FOR AN INDIVIDUAL IS NOT ";
9 PRINT "CONSIDERED SIGNIFICANT. THE TERM 'OVERWEIGHT' IS GENERALLY
  APPLIED TO PERSONS WHO ARE 10-20% ABOVE THE DESIRABLE WEIGHT.
  #OBESITY' IS APPLIED TO PERSONS WHO ARE ABOUT 20% OR MORE OVER-
  WEIGHT. ";
10 PRINT "IN THIS PROGRAM, OBESITY IS DETERMINED BY A CRITICAL
  OBESITY INDEX BASED UPON 'QUETELET'S INDEX' (QI). THIS
  CRITICAL INDEX IS REACHED WHEN THE PERSON'S WEIGHT IS ABOUT
  18% ABOVE THE MAXIMUM IDEAL WEIGHT. ALSO, QI IS USED TO ";
11 PRINT "ESTIMATE BODY FAT (BF). THE BF VIA QI IS IN GOOD
  AGREEMENT WITH THE VALUE FROM TOTAL BODY WATER USING: %BF=100-
  (137* WT. WATER/BODY WT.).": PRINT: PRINT "PRESS 'CONT' TO
  CONTINUE!": STOP: PRINT
12 PRINT "MORE REFERENCES: HUME & WEYERS, J.CLIN.PATH., VOL. 24,
  PP. 234-238 (1971); JAMES, (A DHSS/MRC REPORT) HER MAJESTY'S
  STATIONERY OFFICE, LONDON, 1976 (ISBN 0 11 450034 7). REMARKS
  ARE TO BE FOUND IN LINE #'S 20, 96, 100, 132, 138, 143, 148, 162,
  200, 490."
```

```

13 PRINT: PRINT 'PRESS 'CONT' TO CONTINUE!"; :PRINT: STOP
14 PRINT: PRINT
20 DIM W$(30,5): REM THIS IS AN ARRAY FOR HEIGHT & WEIGHTS
CORRESPONDING TO SMALL, MEDIUM, & LARGE BODY FRAMES
30 FOR J=1 TO 30
40 FOR K=1 TO 4
50 READ W$(J,K)
60 IF W$(J,1) ="ZERO" THEN J=J-1: GOTO 80
70 NEXT K,J
80 PRINT "LIST YOUR SEX (FEMALE=1; MALE=2) ACCORDING TO NUMBERS;
YOUR HEIGHT (NO SHOES) TO NEAREST INCH ";
81 PRINT "(E.G., 5-10); YOUR BODY FRAME(SMALL=1; MEDIUM=2; LARGE=3);
& FINALLY YOUR WEIGHT (IN POUNDS, WITH BED CLOTHING).";

83 INPUT S, H$, F, WT
85 PW=WT
90 ON S GOTO 95,195
95 PRINT: PRINT "-----";
TV=VAL(MID$(H$,1,1))*12+VAL(MID$(H$,3),5)
96 REM LINE #'S 97-99 & 137 EXPRESS THE PROGRAM LIMITATIONS FOR
FEMALES WHILE LINE #'S 197-199 & 237 EXPRESS THE PROGRAM
LIMITATIONS FOR MALES
97 IF F <= 1 AND TV < 60 THEN PRINT: PRINT "BECAUSE YOUR FRAME IS
ONLY 'SMALL' AND YOUR HEIGHT IS LESS THAN 5-0, THIS PROGRAM
CAN OFFER YOU NO INFORMATION!": GOTO 180
98 IF F <= 1 AND TV > 70 THEN PRINT: PRINT "BECAUSE YOUR FRAME IS
ONLY 'SMALL' AND YOUR HEIGHT IS MORE THAN 5-10, THIS PROGRAM
CAN OFFER YOU NO INFORMATION!": GOTO 180
99 IF TV < 60 OR TV > 70 THEN PRINT: PRINT "BECAUSE OF HEIGHT
LIMITATIONS, THIS PROGRAM CANNOT GIVE YOU YOUR IDEAL WEIGHT!":
GOTO 140
100 FOR W = 1 TO 11: REM THIS IS FOR FEMALES ONLY
110 L = 1
120 IF W$(W,L) =H$ THEN PRINT: PRINT "YOUR IDEAL WEIGHT =";
W$(W,F+1);; GOTO 133
130 NEXT
132 REM LINE #'S 133& 233 DETERMINE THE % THAT YOUR WEIGHT
EXCEEDS YOUR MAXIMUM IDEAL WEIGHT
133 SS =VAL(MID$(W$,F+1), 5): IF SS<PW THEN PRINT "----YOU ARE "
INT(((PW-SS)*100/SS)*10+.5)/10" % ABOVE YOUR MAXIMUM IDEAL
WEIGHT. ";
134 GG =INT(((PW-SS)*100/SS)*10+.5)/10: IF GG>10 AND GG<18
THEN PRINT "(YOU ARE CONSIDERED OVERWEIGHT)!"
136 PRINT
137 IF F <= 1 THEN PRINT: PRINT "BECAUSE YOUR FRAME IS ONLY 'SMALL',
ADDITIONAL INFORMATION ON BODY WATER, ETC. DOES NOT APPLY TO
YOU!": GOTO 180
138 REM IN LINE #'S 140 & 240, 'H' DENOTES HEIGHT IN METERS &
'WT' WEIGHT IN KILOGRAMS OF FEMALES & MALES, RESPECTIVELY
140 H =VAL(MID$(H$,1,1))*12 + VAL(MID$(H$,3),5): H =H*.0254:
WT = WT*.45359
143 REM IN LINE #'S 145 & 245, 'QI' DENOTES 'QUETELET'S INDEX'
WHICH IS USED FOR FEMALES & MALES RESPECTIVELY, TO DETERMINE
BODY FAT AND WHETHER A PERSON IS OBESE
145 QI = WT/(H*H)
148 REM LINE #'S 150-160 & 250-260 DETERMINE WHETHER OBESITY
EXISTS FOR FEMALES & MALES, RESPECTIVELY
150 IF F=2 AND QI > 27.0 THEN PRINT: PRINT "YOU ARE ALSO OVER THE
CRITICAL OBESITY INDEX----START DIETING!": GOTO 165
155 IF F=3 AND QI > 29.5 THEN PRINT: PRINT "YOU ARE ALSO OVER THE
CRITICAL OBESITY INDEX----START DIETING!": GOTO 165
160 PRINT: PRINT "YOU ARE UNDER THE CRITICAL OBESITY INDEX---
CONGRATULATIONS!"
162 REM LINE #'S 165 & 265 ALLOW DETERMINATION OF BODY FAT IN
WOMEN & MEN, RESPECTIVELY; #'S 170 & 270 ALLOW DETERMINATION
OF BODY SURFACE AREAS FOR WOMEN & MEN, RESPECTIVELY; #'S 175

```

```

& 275 ALLOW DETERMINATION OF BODY WATER IN WOMEN AND MEN,
RESPECTIVELY
165 WF = 1.48*QI-7: PRINT: PRINT "YOUR BODY FAT = " INT(WF*10+.5)/10
    " %"
170 BS = (.007185*WT^4.25*(H*100)^.725): PRINT: PRINT "YOUR
    BODY SURFACE AREA IS CA. " INT(BS*100 + .5)/100" SQ. METERS
    (OR " INT((BS*10.764)*10 + .5)/10" SQ. FT.)"
175 WW = .183809*WT + 34.4547*H-35.270121
177 PRINT: PRINT "YOUR TOTAL BODY WATER IS CA. " INT(WW*10 + .5)/10
    " LITERS (CA. " INT((WW*1000/453.6)*10 + .5)/10" LBS. WATER
    ---OR " INT((WW*1002/WT)*10 + .5)/10 " % BODY WT.)"
180 PRINT"-----": PRINT: GOTO 80
195 PRINT: PRINT"-----":
    TV = VAL(MID$(H$,1,1))*12 + VAL(MID$(H$,3))
197 IF F <= 1 AND TV < 64 THEN PRINT: PRINT "BECAUSE YOUR FRAME IS
    ONLY 'SMALL' & YOUR HEIGHT IS LESS THAN 5-4, THIS PROGRAM
    CAN OFFER YOU NO INFORMATION!": GOTO 280
198 IF F <= 1 AND TV > 95 THEN PRINT: PRINT "BECAUSE YOUR FRAME IS
    ONLY 'SMALL' & YOUR HEIGHT IS MORE THAN 6-3, THIS PROGRAM
    CAN OFFER YOU NO INFORMATION!": GOTO 280
199 IF TV < 64 OR TV > 75 THEN PRINT: PRINT"BECAUSE OF HEIGHT
    LIMITATIONS, THIS PROGRAM CANNOT GIVE YOU YOUR IDEAL WEIGHT!":
    GOTO 240
200 FOR W = 12 TO 23 : REM THIS IS FOR MALES ONLY
210 L=1
220 IF W$(W,L) = H$ THEN PRINT: PRINT"YOUR IDEAL WEIGHT = ";
    W$(W,F+1); GOTO 233
230 NEXT
233 SS = VAL(MID$(W$(W,F+1),5)): IF SS < PW THEN PRINT"---YOU ARE "
    INT(((PW-SS)*100/SS)*10 + .5)/10 " % ABOVE YOUR MAXIMUM IDEAL
    WEIGHT ";
234 GG = INT(((PW-SS)*100/SS)*10 + .5)/10: IF GG > 10 AND GG < 18
    THEN PRINT"(YOU ARE CONSIDERED OVERWEIGHT)!"
236 PRINT
237 IF F <= 1 THEN PRINT: PRINT"BECAUSE YOUR FRAME IS ONLY 'SMALL',
    ADDITIONAL INFORMATION ON BODY WATER, ETC. DOES NOT APPLY
    TO YOU!": GOTO 280
240 H = VAL(MID$(H$,1,1))*12 + VAL(MID$(H$,3)): H = H*.0254 :
    WT = WT*.45359
245 QI = WT/(H*H)
250 IF F = 2 AND QI > 27.5 THEN PRINT: PRINT"YOU ARE ALSO OVER
    THE CRITICAL OBESITY INDEX---START DIETING!": GOTO 265
255 IF F = 3 AND QI > 29.9 THEN PRINT: PRINT"YOU ARE ALSO OVER
    THE CRITICAL OBESITY INDEX---START DIETING!": GOTO 265
260 PRINT: PRINT"YOU ARE UNDER THE CRITICAL OBESITY INDEX---
    CONGRATULATIONS!"
265 MF = 1.281*QI-10.13 : PRINT: PRINT"YOUR BODY FAT = "
    INT(MF*10 + .5)/10 " %"
270 BS = (.007185*WT^4.25*(H*100)^.725): PRINT: PRINT"YOUR
    BODY SURFACE AREA IS ABOUT " INT(BS*100 + .5)/100 " SQ.
    METERS (OR " INT((BS*10.764)*10 + .5)/10 " SQ. FEET)"
275 MW = .296785*WT + 19.4786*H-14.012934
277 PRINT: PRINT"YOUR TOTAL BODY WATER IS ABOUT " INT(MW*10 + .5)/10
    " LITERS (ABOUT " INT((MW*1000/453.6)*10 + .5)/10 " % BODY WEIGHT)"
    WATER---OR " INT((MW*1002/WT)*10 + .5)/10 " % BODY WEIGHT)"
280 PRINT"-----": PRINT: GOTO 80
490 REM LINE #'S 500-600 CONTAIN HEIGHT-WEIGHT DATA FOR FEMALES
    ONLY WHILE #'S 750-860 CONTAIN HEIGHT-WEIGHT DATA FOR MALES
    ONLY
500 DATA "5-0", "99-107", "104-116", "112-128"
510 DATA "5-1", "102-110", "107-119", "115-131"
520 DATA "5-2", "105-113", "110-123", "118-135"

```


530 DATA "5-3", "108-116", "113-127", "122-139"
 540 DATA "5-4", "111-120", "117-132", "126-143"
 550 DATA "5-5", "115-124", "121-136", "130-147"
 560 DATA "5-6", "119-128", "125-140", "134-151"
 570 DATA "5-7", "123-132", "129-144", "138-155"
 580 DATA "5-8", "127-137", "133-148", "142-160"
 590 DATA "5-9", "131-141", "137-152", "146-165"
 600 DATA "5-10", "135-145", "141-156", "150-170"
 750 DATA "5-4", "118-126", "124-136", "132-149"
 760 DATA "5-5", "121-130", "127-140", "135-153"
 770 DATA "5-6", "125-134", "131-144", "139-158"
 780 DATA "5-7", "129-138", "135-149", "144-163"
 790 DATA "5-8", "133-142", "139-153", "148-167"
 800 DATA "5-9", "137-147", "143-157", "152-171"
 810 DATA "5-10", "141-151", "147-162", "156-176"
 820 DATA "5-11", "145-155", "151-167", "161-181"
 830 DATA "6-0", "149-159", "155-172", "165-186"
 840 DATA "6-1", "153-164", "159-177", "170-191"
 850 DATA "6-2", "157-168", "164-182", "175-196"
 860 DATA "6-3", "161-172", "169-187", "179-201"
 1000 DATA ZERO

PROGRAM EXAMPLES

COMMAND: 'RUN'→ STATEMENTS IN LINE #15 3-6 AND 'PRESS 'CONT'
 TO CONTINUE!
 BREAK IN 7"
 COMMAND: 'CONT'→ STATEMENTS IN LINE #15 8-11 AND 'PRESS 'CONT'
 TO CONTINUE!
 BREAK IN 11"
 COMMAND: 'CONT'→ STATEMENT IN LINE # 12 AND 'PRESS 'CONT' TO
 CONTINUE!
 BREAK IN 13"
 COMMAND: 'CONT'→ "LIST YOUR SEX (FEMALE = 1; MALE = 2) ACCORDING

TO NOS.; YOUR HEIGHT (NO SHOES) TO NEAREST INCH
 (E.G., 5-10); YOUR BODY FRAME (SMALL = 1; MEDIUM = 2;
 LARGE = 3); & FINALLY YOUR WEIGHT (IN LBS. WITH BED
 CLOTHING): ?"
 RESPONSE: '2, 5-8, 2, 165' →
 "-----
 YOUR IDEAL WEIGHT = 139-153---YOU ARE 7.8% ABOVE YOUR
 MAX. IDEAL WT.
 YOU ARE UNDER THE CRITICAL OBESITY INDEX---
 CONGRATULATIONS!
 YOUR BODY FAT = 22%
 YOUR BODY SURFACE AREA IS CA. 1.88 SQ. METERS
 (OR 20.3 SQ. FT.)
 YOUR TOTAL BODY WATER IS CA. 41.8 LITERS (CA. 92.2
 LBS. WATER---OR 55.9% BODY WT.)
 "-----
 LIST YOUR SEX (FEMALE = 1; MALE = 2) ACCORDING TO
 NUMBERS; YOUR HEIGHT (NO SHOES) TO NEAREST INCH
 (E.G., 5-10); YOUR BODY FRAME (SMALL = 1; MEDIUM = 2;
 LARGE = 3); & FINALLY YOUR WEIGHT (IN LBS. WITH
 BED CLOTHING): ? "
 RESPONSE: '1, 5-2, 2, 148' →
 "-----
 YOUR IDEAL WEIGHT = 110-123---YOU ARE 20.3% ABOVE
 YOUR MAXIMUM IDEAL WEIGHT
 YOU ARE ALSO OVER THE CRITICAL OBESITY INDEX---
 START DIETING!
 YOUR BODY FAT = 33.1%
 YOUR BODY SURFACE AREA IS CA. 1.68 SQ. METERS
 (OR 18.1 SQ. FT.)
 YOUR TOTAL BODY WATER IS CA. 31.3 LITERS (CA. 69.1
 LBS. WATER---OR 46.7% BODY WT.)
 "----- "



Skyles Electric Works

PET 2001-8 PET owners:

Do You Want Your PET To Be a Word Processor, Too?

Well, it can be...with Skyles Electric Works' new Word Processing PCB designed especially for the 2001-8. You'll need Commodore's new Disk Drive, of course. And you'll need an additional 8K of RAM memory. (We recommend the Skyles Memory Expansion System, of course.)

Skyles then supplies the interfacing PCB on which you can put the Commodore Word Processor chip. Or, even better, you can buy from Skyles the PCB complete with Commodore's Word Processor in place.

But wait: you can add the Toolkit at the same time; the Toolkit with the so important ten commands. Here's the lineup:

The PCB, to accept the Word Processor only	\$ 30.00*
The PCB, to accept both Word Processor and Toolkit	40.00*
The PCB, to accept Word Processor; Toolkit tested and installed	90.00*
The PCB with the Word Processor tested and installed	140.00*
The PCB with both the Word Processor and Toolkit tested and installed.	190.00*

From S.E.W. only: custom designed for your PET 2001-8 to interface with most memory expansion systems. Or, even better, with the 8KB Skyles Memory Expansion System.

**PET LOVERS SPECIAL:
S.E.W. MEMORY EXPANSION SYSTEMS**

8KB Memory Expansion System \$225
16KB Memory Expansion System \$425
24KB Memory Expansion System \$625

SPECIAL PRICE WITH PURCHASE OF ANY WORD
PROCESSOR OPTION ABOVE

This offer Expires February 14, 1980

2001-8 owners, you can now use your PET for word processing. Skyles Electric Works didn't forget you...

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

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



Skyles Electric Works

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

Lifetime of a Non-Renewable Resource

One of the great problems facing the world today is the conservation of resources, particularly those which can not be readily renewed. The simple program is a good model of an interactive BASIC simulation.

Marvin L. DeJong
Dept. of Math & Physics
The School of the Ozarks
Point Lookout, MO 25726

Are you interested in doing something simple, serious, and of educational value with your computer? Estimating the lifetime of a non-renewable resource such as coal, oil, or natural gas is often a difficult calculation involving calculus and the use of exponential or logarithmic functions. The computer makes it short and super simple, as you will see. The results have serious implications. An editorial in our local paper claimed that we have enough coal to last for centuries. This may or may not be true. Read on.

Suppose there are R tons of coal still unmined. Also suppose that we use C tons of coal per year. At the end of one year we will have $R - C$ tons left. The next year we subtract C tons again, and so on until our coal is gone. If we kept track of the number of subtractions, we would know how many years the coal would last. This is the lifetime of the resource.

However, we must take into account that, typically, the production and consumption of resources increases over time. Our demand for electrical power, fuel oil, natural gas, and gasoline grows. The gross national product, or GNP, increases in a healthy (?) economy. Growth implies increases in the consumption of resources, and this must be taken into account when calculating the lifetime of a resource.

Assume that consumption of a resource grows by G percent per year. If C tons of coal are consumed this year, then next year we will consume C tons of coal plus the increase, which is $G/100$ multiplied times C . Anyone who has calculated interest compounded annual-

ly knows how to do the arithmetic. A simple example may help. If we use 500 million tons of coal this year, and our growth rate in the consumption of coal is 10 per cent per year, then next year we will consume 500 million tons plus 10 per cent of 500 million tons.

The calculation of the lifetime of a resource is much the same as outlined above, except that C increases by G percent each year if the growth factor is taken into account. A flowchart of the entire process is shown in Figure 1, and

```
10 PRINT "THIS IS A PROGRAM TO CALCULATE HOW LONG A NON-RENEWABLE
    RESOURCE WILL LAST."
20 PRINT "TYPE IN THE ESTIMATED RESERVES OF THE RESOURCE."
30 PRINT "RESERVES=";
40 INPUT R
50 PRINT "TYPE IN THE ANNUAL RATE AT WHICH THE RESOURCE IS CONSUMED."
60 PRINT "CONSUMPTION RATE=";
70 INPUT C
80 PRINT "TYPE IN THE ANNUAL PERCENT INCREASE IN THE CONSUMPTION RATE."
90 PRINT "GROWTH RATE OF CONSUMPTION=";
100 INPUT G
110 G=G/100
120 Y=0
130 R=R-C
140 C=C+C*G
150 Y=Y+1
160 IF R > 0 THEN 130
170 PRINT "YOUR RESOURCE WILL LAST"; Y; "YEARS."
180 END
```

Table 1: Resource Depletion Program

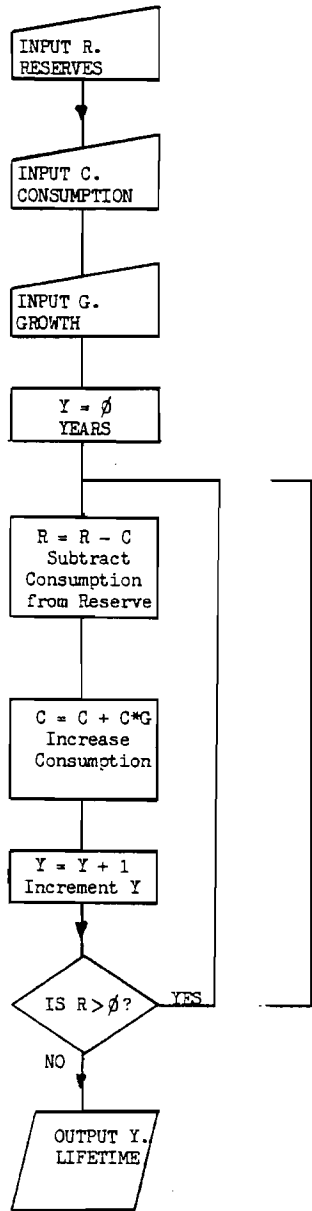


Figure 1: Flowchart to calculate the Lifetime of a Non-Renewable Resource

Resource	Reserves	Current Consumption Rate	Growth
U.S. Coal	500 billion tons	0.7 billion tons per yr.	7%
U.S. Oil	100 billion barrels	6 billion barrels per yr.	8%
World Oil	2000 billion barrels	20 billion barrels per yr.	7%

Table 2: Data on Reserves, Consumption & Growth of 2 Resources

the corresponding BASIC program is given here. There is only one approximation in the calculation. I assumed that the consumption changes abruptly at the end of a year, whereas it actually changes more or less continuously throughout the year, much like interest compounded daily. The approximation has little effect on the results. The error is usually less than a year or two.

Next we need some data to put into the computer. This can be a bit tricky because companies that sell the resource tend to overestimate the reserves, while conservationists are probably biased in the other direction. The truth is most likely somewhere in the middle. My data (and the inspiration for this project) came from an article by Professor Albert A. Bartlett in the September 1978 issue of the *American Journal of Physics*. The data, which I have taken the liberty to round off to one significant digit, appears in Table 1. Other references you might want to check are: Dr. M. King Hubbert, *A National Fuels and Energy Policy Study*, Serial 93-40 (92-75) Part I, U.S. GPO, Washington, D.C., 1973, \$2.35; and Dr. M. King Hubbert, "Energy Resources of the Earth" in *Scientific American*, September 1971.

Almost any computer should take the simple BASIC program given here. Mine ran on my Microsoft BASIC for the KIM-1. Since almost every step is illustrated in the flowchart, no further explanation of the program is necessary. Load it and type in the data as they are requested. When the last item is entered, hit RETURN and wait for the answer.

Now experiment with the input data. Suppose the estimate of the reserve was half as large as it really is. How does this change the lifetime of the resource? Does doubling the reserve double the lifetime? Calculate the lifetime with a 0 per cent growth rate; a 10 per cent growth rate. Get data on natural gas, copper, or other non-renewable resources and run the program. What are the actual conditions under which coal will last for centuries?

Classified Ads

MEDICAL COMPUTER JOURNAL
Subscribe to a Journal dedicated entirely to the use of computers in clinical practice: original articles, programs and reports software. Today membership \$15.00.

A. Ghaussey, M.D.
Medical Computer Journal
42 E. High Street
E. Hampton, CT 06424
tel. (203) 267-2934

APPLESOFT UTILITIES
Optimization Library \$20
Structured BASIC \$25

APPLESOFT GAMES:
HIRES Lunar Lander \$12 (24K)
INTEGER BASIC GAMES
2-player Master Maze \$12 (16K)
On disk or tape from:

SENSIBLE SOFTWARE
P.O. Box 2395
Dearborn, MI 48123

"Real" Tennis for APPLE II 16K
TELE-TENNIS: a sophisticated paddle game for two players. Side view game features scoring, numerous serves, volleys, lobs, etc. "Gravity" based. Ideal gift! \$9.95; checks only.

A2Devices
1970-D Ballena Blvd.
Alameda, CA 94501
tel. (415) 763-4064

PET CONTEST!!! \$10.00 prize
Mach. L. Othello—must beat 5 times and doc. Multiply search by D. Schaeckter \$15.00 entry fee.

J.K. Johnson
9304 Emory Grv. Rd.
Gaithersburg, MD 20760

Challenger (25—Matches) A full screen video graphics and sound game. Play your computer or a friend! For OSI C—1P and C—2P, specify system. 300—Baud cassette and documentation. Only \$5.00 post paid from:

DWA Software
407 Hotcrest
Midland, MI 48640

OSI Software—Cursor 2(back space and clear screen from keyboard) for all comp. with DIC/WM on break \$5.95. Chess (full graphics) \$19.95. Word Process. (cassette) \$8.95. Backgammon \$9.95. (3) card game pack \$15.95. Utility prog's, memory maps, etc. Catalogue \$1.00.

J. Endic/Progressive Computing
3336 Avondale Ct. Windsor,
Ontario, Canada, N9E1X6

More on page 37.

The Loneliness of the Microcomputer

While most of us would agree that the microcomputer is a pretty great device, it is not without potential problems. One of the possible drawbacks to the microcomputer which I have not seen discussed is that of its almost exclusive "one-on-one" utilization. Much has been said about this type of problem with television. Instead of getting together with friends, family or neighbors after dinner, how many people now just sit in front of the "boob tube"? How much human interaction has been given up in order to watch TV?

Microcomputers seem to be used in a mode very similar to TV watching. One person interacts with the microcomputer. Other people are not required and, unless you are showing off your latest program, are generally not wanted! Hardly a socialable device. Think about the things you do with your micro. How many of them involve another human? Balancing your check book, playing chess or life, solving equations ... the list goes on. Most of the programs which have been listed in the Micro Software Catalog and many of the programs presented in articles have been of the single individual variety.

Assuming you agree that it would be nice to make the micro more socialable, how can this be done? Some micro uses are inherently individual. You do not necessarily want a friend

helping you balance your checkbook. Other areas can be modified to permit multi-individual use and interaction. The entire games area is open to the generation of games which several people play, not just one. In a multi-person game, the micro can be used to generate and maintain a very complex playing situation, can generate sophisticated environments and display them in a variety of forms, can be the score keeper and when necessary the arbiter, can inform and assist the players, can be a time keeper, and so forth. The micro is this type of game is not the opponent. I hope that we will see more games of this type in the near future.

Other multi-person micro applications are starting to appear. A number of systems are being set up which permit individuals to communicate with one another through their micros. There should be other areas developed which permit the multi-person utilization of micros. I feel that it is important for every computerist to occasionally question how he is using his equipment, and to determine what the secondary effects of the uses may be.

Robert M. Trapp



**PO Box 6502
Chelmsford, Mass 01824
617-256-5515**

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

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

Back Issues:

Issues 7 to 12:
Issues 13 on:

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

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

Name:

Address:

City: State: Zip:

Country (if not U.S.):

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

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

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

Microcomputer Usage: Educational Business Personal Control Games Other:

Languages Used: Assembler BASIC FORTH PASCAL Other:

Your comments and suggestions on MICRO:

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

Surface:

United States \$15.00
All Other Countries \$18.00

Air Mail:

Central America \$27.00
Europe/So. America \$33.00
All Other Countries \$39.00 \$

"BEST of MICRO Volume 1"

Surface \$7.00
Air Mail \$10.00 \$

"BEST of MICRO Volume 2"

Surface \$9.00
Air Mail \$13.00 \$

No. Surface @ \$1.75 each = \$

No. Air Mail @ \$2.75 each = \$

No. Surface @ \$2.25 each = \$

No. Air Mail @ \$3.25 each = \$

TOTAL \$

EASYWRITER,* the 1st true Word Processor for the Apple!*

Are you looking for the best Word Processor for your Apple? Well we are so sure you'll choose **EasyWriter** that we've prepared this ad to help you make your decision **EASY**. Check out these powerful features:

- Incremental spacing to support your Qume, Diablo, or Spinwriter
- Character oriented (No line numbers to deal with)
- Menu selectable routines for all known printers and interfaces
- Word wrap around on screen for continual text entry
- Our own new high speed DOS (Twice as fast as Apple's)
- Of course full editing, disk, and printer commands
- Subscripting, Superscripting, and MORE MORE MORE . . .

The straight facts make EasyWriter the only logical choice. By the same people who brought you WHATSIT. Available at your local computer store or our new California office!

**793 Vincente
Berkeley, CA 94707
(415) 525-4046**

**146 N. Broad Street
Griffith, IN 46319
(219) 924-3522**

It Isn't Software Until it Works!

***EasyWriter**

EasyWriter is a TM of Cap'n Software

***Dr. Memory**

Dr. Memory is a TM of Muse

***Big Edit**

Big Edit is a TM of Gravey, Martin & Sampson, Inc.

***Apple Pie**

Apple Pie is a TM of Programma International, Inc.

***Super-Text**

Super-Text is a TM of Muse

Apple

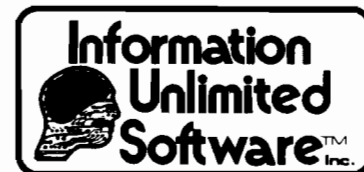
Apple is a TM of Apple Computers, Inc.

Whatsit

Whatsit is a TM of Computer Headware

	<i>EasyWriter*</i>	<i>Dr. Memory*</i>	<i>Big Edit*</i>	<i>Apple Pie*</i>	<i>Super-Text*</i>		
Incremental Spacing	X						
Character Oriented Screen	X				X		
Printer Interface Routines	X						
Word Wrap Around	X				X		
Editing Commands	X	X	X	X	X		
High Speed Disk	X						
50 Pages of Text per Disk	X						
Subscripting & Superscripting	X				X		
Your Choice							

ORDER FORM



CUT ALONG DOTTED LINE

Name: _____

Company: _____

Address: _____

City: _____ State: _____ Zip: _____

I would like more information on:

ITEM	QUANTITY	PRICE	TOTAL
Easy Writer (EZ 2)		\$ 99.95	
Whatsit Model A-1 (Apple)		125.00	
Whatsit Model CP-2 (CP/M)		150.00	
Whatsit Model NS-3 (N Star)		100.00	
Whatsit Manual		25.00	
EasyWriter Manual		30.00	
Subtotal			\$
CA Residents add 6% sales tax			\$
Shipping & Handling			\$ 2.50
GRAND TOTAL			\$

Master Charge or Visa Number: _____

Sweet-16 Programming Using Macros

Some very useful information is presented about Macros in general, the APPLE II Sweet-16 Interpreter, and how to use them together.

Richard C. Vile, Jr.
3467 Yellowstone Drive
Ann Arbor, MI 48105

The history of computer programming is replete with stories of the development of new tools. Assemblers were designed with the purpose of relieving the programmer of the tedium of programming in binary machine language. Over the course of the past twenty years, various features have been added to assembly languages to further ease the pain. Prime among these inventions has been the macro capability available in many assemblers. Macros provide means for extending the expressive capabilities of assembly language. Another software tool developed in recent years is the virtual machine. A virtual machine is emulated, imitated or interpreted by a program. It provides capabilities not directly available in the hardware of the real machine on which it is simulated. This article discusses the combined application of macro assembly and virtual machine interpretation on the APPLE II personal computer system.

Macro Assemblers

Macro assemblers extend the capabilities of ordinary assemblers by providing ways to abbreviate commonly used sequences of instructions. Often a programmer will use sequences of instructions that have identical opcodes and addressing modes, but differ only in the memory locations referred to. Consider the following:

```
INC LOC1L  
BNE = +5  
INC LOC1H
```

and

```
INC LOC2L  
BNE = +5  
INC LOC2H
```

where the symbol '=' is used to refer to the location of the instruction being assembled. These two sequences both have the same purpose: to cause the 16 bit quantity stored in two consecutive memory locations to be increased by one. For this example we have assumed that the locations are not in page zero and are directly addressed. A macro assembler will allow these sequences to be abbreviated using a new symbol, chosen by the programmer. The symbol must be formally declared in a *Macro Definition*, before it is used. Such a definition is shown below using the notation of the ASM/TED assembler of Carl Moser:

```
!!!! INCD .MD (WHERE)  
INC WHERE  
BNE = +5  
INC WHERE + 1  
.ME
```

The symbol WHERE does not represent a specific memory location, but potentially many different memory locations.

It is called, in assembler terminology, a *formal or dummy parameter*. Even though our example has only one formal parameter, macros in general may have many. The three exclamation marks preceding the name INCD indicate to the assembler that the label INCD is the name of a macro. '.MD' stands for *Macro Definition* and '.ME' stands for *Macro End*. The sequence of instructions between .MD and .ME is called the *body* of the macro. Once a macro definition is written into a program, the macro may subsequently be *called* by using its name in an instruction, as if it were an opcode. More sophisticated macro assemblers allow macros to appear in any field of an instruction, rather than just the opcode field. When a macro is called, the programmer is obligated to supply *actual* parameters to replace the dummy parameters used in the definition. In the example given above, when INCD is called, it must be accompanied by the label associated with an actual memory location used by the program:

```
INCD (COUNT)
```

The actual parameter is substituted for all occurrences of the dummy parameter in the macro body and the instructions in the macro body are assembled directly into the program at the point of the macro call. This is known as "expanding" the call:

```

INC COUNT
BNE = +5
INC COUNT+1

```

Another way of thinking about macros is to view them as small subroutines which are inserted directly into a program instead of being called. When a short sequence of instructions is commonly repeated, it may be cheaper to make a macro out of it than to make it into a subroutine. Part of the reason for this is that it costs extra instructions to pass parameters to a subroutine, especially on a micro such as the 6502, which has a limited number of registers. In this example, particularly, the difference is significant. In order to convert the INCD macro into a subroutine, we would need to figure out a way to pass the address of the first byte to be incremented. For example:

```

L D A "Low byte of address of
      COUNT"
L D X "High byte of address
      of COUNT"
J S R INCD
.
.
.
I N C D S T A CL Page Zero Loc
      S T X CH Next Page Zero Loc
      L D Y #00 Assuming Y
      ; available- otherwise
      ; TAY-PHA first.
      I N C (CL),Y
      B N E = +6
      I N Y
      I N C (CL),Y
      R T S

```

This is surprisingly more complicated than the macro, which is why you probably never thought of making it into a subroutine before. In general, if a subroutine is short and if it involves manipulating addresses of parameters, then it may be worth converting to a macro.

Assemblers vary widely in the richness of features supported. One of the more desirable features to use in conjunction with the macro capability is that of *conditional assembly*. This enables a program to define instruction sequences and, in particular, macros, with much more flexibility. We shall see this in action when we discuss the Sweet-16 macros later. Conditional assembly directives allow the programmer to control the actions taken by the assembler.

Macros can be used to generate arbitrary bit patterns into the stream of object code produced by the assembly of a program. There may be subtle reasons for wishing to do this. One of those reasons forms the meat of our principal example: the bit patterns so generated

may form interpretive code which can, via the macro capability, be interspersed with ordinary machine code. By using macros to generate the interpretive code, the programmer is freed from the odious task of hand assembly — a task which could discourage him from using the interpretive code in the first place.

Sweet-16

The 6502 microprocessor provides no direct capability for handling 16 bit quantities. In particular, the machine has no internal 16 bit registers, save for the PC. Thus, when it becomes necessary to do 16 bit arithmetic, or to manipulate pointers or 16 bit addresses, the programmer is forced to write instruction sequences to simulate the required operations. The APPLE II firmware contains a subroutine known as the SWEET-16 "dream machine," which does just that. It operates in an interpretive mode, taking the sequence of bytes following the instruction which calls it as virtual or interpretive code. Here's how it works.

When a JSR (*Jump to SubRoutine*) instruction is executed by the 6502 processor, the value of the program counter, which in that case will be the address of the last byte of the JSR instruction, is saved on the 6502 stack as two consecutive bytes. When a RTS

(*ReTurn from Subroutine*) instruction is executed within the called subroutine, that address increased by one will be restored from the stack to the PC, to enable the 6502 to continue executing instructions following the JSR instruction. (See Figure 1.) The fact that the "return" address is saved on the stack means that the called subroutine can, in fact, find out where it was called from. More than that, it can use the return address and the indirect addressing mode of the 6502 to actually retrieve the sequence of bytes following the calling instruction. That is precisely what the Sweet-16 subroutine does.

The Sweet-16 interpreter takes advantage of the fact that the return address is at the top of the 6502 stack. It pops the two bytes from the stack and transfers them to a pair of page zero locations which it then uses as an indirect address to locate the sequence of interpretive instructions following the JSR which called it.

Thus the return address of the Sweet-16 subroutine becomes the address of the first instruction to be executed by the Sweet-16 machine. As the Sweet-16 machine executes instructions, it updates this address to point to the next virtual instruction to be executed. When the Sweet-16 interpreter finds an interpretive instruction

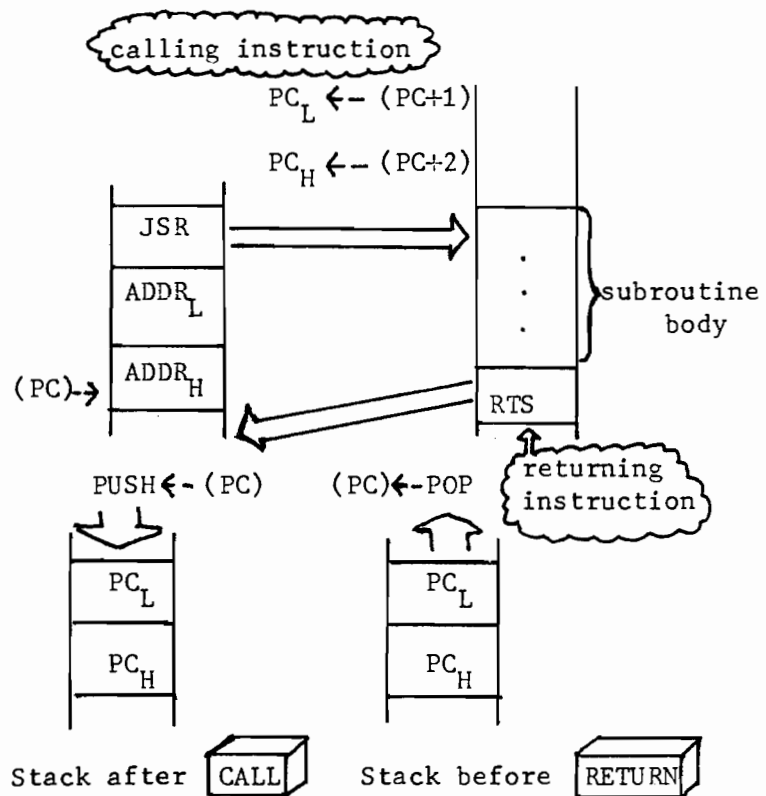


Figure 1: 6502 Subroutine Call and Return

"return," it restores the address of that instruction to the stack and executes a real 6502 RTS. This causes the processor to continue execution of the machine code following. Thus, Sweet-16 code and real 6502 code may be mixed together in sequence, with Sweet-16 being called by a JSR instruction preceding each "chunk" of Sweet-16 code.

The Sweet-16 processor contains 16 registers, each simulated by two page zero locations. Register 15 doubles as the Sweet-16 program counter. As explained above, the actions of the various Sweet-16 instructions cause the contents of the virtual PC to be updated. The cycle of execution of the Sweet-16 machine is:

1. Fetch Opcode LDY #00
 LDA (R15),Y
2. Execute Opcode Transfer control to the appropriate section of Sweet-16.

Op	Mnemonic	Arguments	Effect
1n	SETR	(Rn,Constant)	Rn ← Constant
2n	LD	(Rn)	R0 ← Rn
3n	ST	(Rn)	Rn ← R0
4n	LD@	(Rn)	High byte of R0 ← 0 Low byte of R0 ← (Rn)
5n	ST@	(Rn)	(Rn) ← Low byte of R0
6n	LDD@	(Rn)	R0 _L ← (Rn); Rn ← Rn+1; R0 _H ← (Rn)
7n	STD@	(Rn)	(Rn) ← R0 _L ; Rn ← Rn+1; (Rn) ← R0 _H
8n	POP@	(Rn)	Rn ← Rn-1; R0 _L ← (Rn); R0 _H ← 0
9n	STP@	(Rn)	Rn ← Rn-1; (Rn) ← R0 _L ; Rn ← Rn-1; (Rn) ← R0 _H
An	ADD	(Rn)	R0 ← R0 + Rn
Bn	SUB	(Rn)	R0 ← R0 - Rn
Cn	POPD@	(Rn)	Rn ← Rn-1; R0 _L ← (Rn); Rn ← Rn-1; R0 _H ← (Rn)
Dn	CPR	(Rn)	Set branch conditions as a result of R0 - Rn. Store result into R13.
En	INCR	(Rn)	Rn ← Rn + 1
Fn	DECR	(Rn)	Rn ← Rn - 1
00	RTN		Return to caller
01	BR	(addr)	Relative branch to addr. (Note: Argument is assembled as displacement. Source argument is absolute.)
02	BNC	(addr)	Branch if No Carry
03	BC	(addr)	Branch if Carry
04	BP	(addr)	Branch if prior result Plus
05	BM	(addr)	Branch if prior result Minus
06	BZ	(addr)	Branch if prior result Zero
07	BNZ	(addr)	Branch if prior result Non Zero
08	BML	(addr)	Branch if prior result = -1
09	BNML	(addr)	Branch if prior result ≠ -1
0A	BK		Execute 6502 Break instruction
0B	RS		Return from Sweet-16 subroutine
0C	BS	(addr)	Branch to Sweet-16 subroutine. addr must be in the range allowable for a relative branch. Return address is stored in a pseudo-stack whose address is contained in R12.

Table 1: Sweet-16 Instruction Set Summary

3. Repeat at 1. or Return to caller (if interpretive opcode was "return").

The following table briefly summarizes the opcodes which the Sweet-16 machine provides. The mnemonics used in the table are those chosen for the macro implementation discussed below. Further details and some examples may be found in the November 1977 issue of BYTE magazine.

The Macros: How They Work

Listing 1. shows the Sweet-16 macros as defined for the Carl Moser ASM/TED macro assembler. The macros fall into two groups: the register and the non-register opcodes. The register opcodes are all assembled to values with a non-zero (\$1 to \$F) high nibble: e.g. LD@(R12) —\$4C. The non-register opcodes all have a 0 value in the high nibble of the opcode byte. Most of the non-register opcodes have a second byte which indicates a relative branch

displacement value, in the style of the 6502 itself. The most interesting part of the set of macro definitions involves the calculation of this displacement. Since the concept of relative branch displacement is common to all the branching operations, a separate macro is defined which is used to calculate the displacement. This macro then gets called in the body of each branching opcode to provide the desired value:

```
!!!RELBR .MD (LOC)
          IF P = -LOC
          .BY LOC - = -1
          ***
          IF M = -LOC
          .BY = -LOC + 1
          ***
          .ME

!!!BR .MD (WHERE)
       .BY 1
       RELBR (WHERE)
       .ME
```

The RELBR macro uses the conditional assembly features of the macro assembler. Let us examine it line by line:

```
!!!RELBR .MD (LOC)
```

This line indicates to the assembler that a Macro Definition is being initiated. The name by which the macro may subsequently be called is RELBR and the argument which must be supplied when it is called is represented by the dummy symbol LOC. When the macro is expanded by a call, the actual argument which is supplied in the call will be substituted for each occurrence of 'LOC' in the body of the definition.

```
IF P = -LOC
```

This line contains one of the conditional assembly operations or *directives* of the assembler: IFP. The assembler is directed to evaluate the expression contained in the remainder of the line; in this case " $= -LOC$ ". If the result is a positive number (the mnemonic stands for *IF Positive*), then the assembler will assemble all instructions following the current line until it encounters a line containing *******, which indicates the end of the scope of the IFP directive. If the expression evaluates to a negative number or zero, then the assembler will ignore all instructions following the current line until the matching *******.

The expression ' $= -LOC$ ' is computed by subtracting the value of the actual parameter substituted for LOC in the call from the value of the assembler's *location counter*, represented in ASM/TED by the

character '='. The location counter represents the address of the instruction being assembled.

B Y LOC— = -1

The directive .BY instructs the assembler to evaluate the expression following and to assemble a single BYte of code from the resulting value. The expression LOC— = -1 computes a value which is the distance from the symbol referenced by 'LOC' to the current location in the object code. This value is converted by the expression to a negative number and adjusted by 1 to account for the fact that the current byte of object has not yet been emitted by the assembler. Note that there is a bug in the definition: if the value LOC— = -1 is less than -128 then an erroneous value will be assembled. This means that the user of the macro set is responsible for avoiding relative branches that are out of range. Note also that the values computed by expressions are in 16 bit, two's complement representation. If such a value is assembled using a .BY directive, the assembler will use the least significant 8 bits (low byte) of the result.

* * *

This line marks the end of the scope of the IFP conditional assembly directive used earlier.

IFM = -LOC

This line has the same intention as the IFP line, except that it tests the result of the expression '= -LOC' for a negative or Minus value. It then does or does not assemble the instructions following the IFM line and up to the matching '*', depending on the outcome of the evaluation.

.BY = -LOC + 1

These instructions are analogous to the corresponding instructions following the IFP directive. The reason for using *both* an IFP and an IFM directive is that the label or location referenced by the dummy argument 'LOC' may turn out to be either ahead of (minus result for = -LOC) or behind (positive result for = -LOC) the instruction which invokes the RELBR macro.

The remainder of the macro definitions are simple and straightforward. A couple of points to note are:

.Defining @SW16 as JSR SW16 makes the macro @SW16 looklike a "new" assembler directive. It says:
Please switch to Sweet-16

.Arithmetic may be performed on dummy arguments:

```
!!!LD .MD (REG)
      .BY $20 + REG
      .ME
```

This fact is crucial to the success of the macros.

Sample Sweet-16 Program

The following program allows the second text page of APPLE II memory to be copied into the first text page. The assembled code is shown to the left.

```
20 89 F6 @SW16
15      SETR (5 $800)
00 08
14      SETR (4 $BFF)
FF 0B
16      SETR (6 $400)
00 04
```

```
45 MOVE LD@ (5)
56     ST@ (6)
24     LD (4)
D5     CPR (5)
04     BP (MOVE)
FA
00     RTN
```

Sweet-16 can also be used more conveniently with this set of macros. They make the assembly source easier to read, and remove the burden of hand assembly from the Sweet-16 programmer.

The reader is urged to learn more about the macro capabilities of assemblers and the labor-saving uses to which they may be applied.

Listing 1.

```
0002 R0      .DE 0
0003 R1      .DE 1
0004 R2      .DE 2
0005 R3      .DE 3
0006 R4      .DE 4
0007 R5      .DE 5
0008 R6      .DE 6
0009 R7      .DE 7
0010 R8      .DE 8
0011 R9      .DE 9
0012 R10     .DE 10
0013 R11     .DE 11
0014 R12     .DE 12
0015 R13     .DE 13
0016 R14     .DE 14
0017 R15     .DE 15
0018        .ES
0019 !!!SETR .MD (REG ADDR)
0020        .BY $10+REG
0021        .SE ADDR
0022        .ME
0023 !!!LD   .MD (REG)
0024        .BY $20+REG
0025        .ME
0026 !!!ST   .MD (REG)
0027        .BY $30+REG
0028        .ME
0029 !!!LD@  .MD (REG)
0030        .BY $40+REG
0031        .ME
0032 !!!ST@  .MD (REG)
0033        .BY $50+REG
0034        .ME
0035 !!!LID@ .MD (REG)
```

0036	.BY \$60+REG	0091	.ME
0037	.ME	0092 !!!BM	.MD (WHERE)
0038 !!!STD@	.MD (REG)	0093	.BY 5
0039	.BY \$70+REG	0094	RELBR (WHERE)
0040	.ME	0095	.ME
0041 !!!POPE	.MD (REG)	0096 !!!BZ	.MD (WHERE)
0042	.BY \$80+REG	0097	.BY 6
0043	.ME	0098	RELBR (WHERE)
0044 !!!STP@	.MD (REG)	0099	.ME
0045	.BY \$90+REG	0100 !!!BNZ	.MD (WHERE)
0046	.ME	0101	.BY 7
0047 !!!ADD	.MD (REG)	0102	RELBR (WHERE)
0048	.BY \$A0+REG	0103	.ME
0049	.ME	0104 !!!BM1	.MD (WHERE)
0050 !!!SUB	.MD (REG)	0105	.BY 8
0051	.BY \$B0+REG	0106	RELBR (WHERE)
0052	.ME	0107	.ME
0053 !!!POPD@	.MD (REG)	0108 !!!BNM1	.MD (WHERE)
0054	.BY \$C0+REG	0109	.BY 9
0055	.ME	0110	RELBR (WHERE)
0056 !!!CPR	.MD (REG)	0111	.ME
0057	.BY \$D0+REG	0112 !!!BRK	.MD
0058	.ME	0113	.BY \$A
0059 !!!INCR	.MD (REG)	0114	.ME
0060	.BY \$E0 + REG	0115 !!!RS	.MD
0061	.ME	0116	.BY \$B
0062 !!!DECR	.MD (REG)	0117	.ME
0063	.BY \$F0+REG	0118 !!!BS	.MD (WHERE)
0064	.ME	0119	.BY \$C
0065 !!!RTN	.MD	0120	RELBR (WHERE)
0066	.BY 00	0121	.ME
0067	.ME	0122 !!!@SW16	.MD
0068 !!!RELBR	.MD (LOC)	0123	JSR \$F689
0069	IFM ==-LOC	0124	.ME
0070	.BY LOC== -1	0999	.EN
0071	***		
0072	IFM ==-LOC		
0073	.BY ==-LOC+1		
0074	***		
0075	.ME		
0076 !!!BR	.MD (WHERE)		
0077	.BY 1		
0078	RELBR (WHERE)		
0079	.ME	/R0=0000	/R1=0001
0080 !!!BNC	.MD (WHERE)	/R3=0003	/R4=0004
0081	.BY 2	/R6=0006	/R7=0007
0082	RELBR (WHERE)	/R9=0009	/R10=000A
0083	.ME	/R12=000C	/R13=000D
0084 !!!BC	.MD (WHERE)	/R15=000F	
0085	.BY 3	//0000,0200,0200	
0086	RELBR (WHERE)		/R2=0002
0087	.ME		/R5=0005
0088 !!!BP	.MD (WHERE)		/R8=0008
0089	.BY 4		/R11=000B
0090	RELBR (WHERE)		/R14=000E

LABEL FILE: [/ = EXTERNAL]

Screen Write/File Routine

Here is a routine, both useful and instructive, which makes it simple to Edit the Apple Screen and Save the Screen Image on Disk.

B.E. Baxter
6761 King's Harbor Drive
Rancho Palos Verdes, CA 90274

The screen write/file routine is a simple 73-byte device to take control away from the monitor and write directly to the screen. All of the escape editing capabilities are supported, so that it is very easy to enter and modify up to and including 21 lines of text. It is equally easy to then save the screen image to disk after completion of text entry.

The source code is straightforward and makes liberal use of monitor routines. Upon entry the cursor is homed and placed on line 1 (not zero). The block labeled KEY continually polls the keyboard and outputs characters through COUT (VIDOUT [\$FBFD] could also be used if printer services are not wanted). The limited editing facilities of the monitor are invoked by typing (escape) followed by one of the command characters. Keyboard entry of (control) Q is used to exit the routine and return to BASIC via \$3D0. Automatic exit is also obtained at line 22. Upon exit, the bell will sound and the BASIC prompt character will appear with the file parameters displayed at the end of the line. At this point the file must be saved using the command, (BSAVE File name) A\$0400, L\$03CF (RETURN). The parenthetical expressions must be typed by the user; that is, type BSAVE file name,

then trace over the remainder of the line with the right arrow to place it into the keyboard buffer and at the end of the line press RETURN. Although I do not find it necessary, a monitor MOVE to page 2 could be set up and inserted at line 225 of the source listing. This would provide back-up in case the BSAVE command is messed up. The object code is assembled at \$0350 and is \$49 bytes long.

In summary, the usage commands are:

Entry to Routine

From BASIC	Call 848
From Monitor	\$0350G

Exit to BASIC Mode

User	(Control) Q
Automatic	Line 22

Edit Screen (See APPLE Ref. Materials) (Escape)

@:	Home cursor (Clear text)
A:	Advance cursor
B:	Backspace cursor
C:	Move cursor down 1 line

D:	Move cursor up 1 line
E:	Clear from cursor to end of line
F:	Clear from cursor to end of screen

Save Screen Image

[BSAVE file name]A\$0400,L\$03CF[CR]

[] = typed by user

Of course it doesn't make much sense to idly write to the screen without some useful purpose. I use the routine to create instruction and documentation files. These files are especially valuable for object code utilities by providing ready access to usage and entry point information. Once the file has been created, it can be handled just like any other file. BLOADing (file name) will immediately display its contents on the screen without requiring any otherwise useful memory. Instruction/print statements in BASIC programs can therefore be eliminated; to be replaced by deferred execution BLOAD disk commands for a very efficient use of main memory.

0100:	0350			ORG	\$0350			
0110:	0350			COUT	*	\$FDED		
0120:	0350			HOME	*	\$FC58		
0130:	0350			CV	*	\$0025		
0140:	0350			TABV	*	\$FB5B		
0150:	0350			RDCHAR	*	\$FD35		
0160:	0350			CROUT	*	\$FD8E		
0170:	0350			BELL	*	\$FF3A		
0180:	0350			POS	*	\$0009		
0190:								
0200:	0350	20	58	FC	JSR	HOME		
0210:	0353	20	8E	FD	JSR	CROUT		
0220:								
0230:	0356	20	35	FD	KEY	JSR	RDCHAF	
0240:	0359	C9	91		CMPIM	\$91		
0250:	035B	F0	0C		BEQ	QUIT		
0260:	035D	A6	25		LDXZ	CV		
0270:	035F	E0	16		CPXIM	\$16		
0280:	0361	F0	06		BEQ	QUIT		
0290:	0363	20	ED	FD	JSR	COUT		
0300:	0366	4C	56	03	JMP	KEY		
0310:								
0320:	0369	A9	16		QUIT	LDAIM	\$16	
0330:	036B	85	25		STAZ	CV		
0340:	036D	20	5B	FB	JSR	TABV		
0350:	0370	20	3A	FF	JSR	BELL		
0360:	0373	A9	E4		LDAIM	\$E4		
0370:	0375	85	09		STA	POS		
0380:	0377	A9	07		LDAIM	\$07		
0390:	0379	85	0A		STA	POS	+01	
0400:	037B	A0	00		LDYIM	\$00		
0410:								
0420:	037D	B9	8A	03	OUT	LDAY	DATA	
0430:	0380	91	09		STAIY	POS		
0440:	0382	C8			INY			
0450:	0383	C0	0F		CPYIM	\$0F		
0460:	0385	D0	F6		BNE	OUT		
0470:	0387	20	D0	03	JSR	\$03D0		
0480:								
0490:	038A	A0			DATA	=	\$A0	
0500:	038B	C1				=	\$C1	A
0510:	038C	A4				=	\$A4	\$
0520:	038D	B0				=	\$B0	0
0530:	038E	B4				=	\$B4	4
0540:	038F	B0				=	\$B0	0
0550:	0390	B0				=	\$B0	0
0560:	0391	AC				=	\$AC	,
0570:	0392	CC				=	\$CC	L
0580:	0393	A4				=	\$A4	\$
0590:	0394	B0				=	\$B0	0
0600:	0395	B3				=	\$B3	3
0610:	0396	C3				=	\$C3	C
0620:	0397	C6				=	\$C6	F
0630:	0398	A0				=	\$A0	
ID=								



NIBBLE is an unusual new Newsletter for Apple II Owners. Each Issue will follow a major theme... such as:

- * DATA BASE MANAGEMENT
- * PROGRAMS FOR THE HOME
- * TEXT PROCESSING
- * COMPUTING FOR KIDS
- * SMALL BUSINESS JOBS
- * GAMES AND GRAPHICS
- * PRACTICAL PASCAL
- * etc.

Significant programs will be in each issue, surrounded by articles which show how to USE the programming ideas in your OWN programs.

Examples of Upcoming Articles...

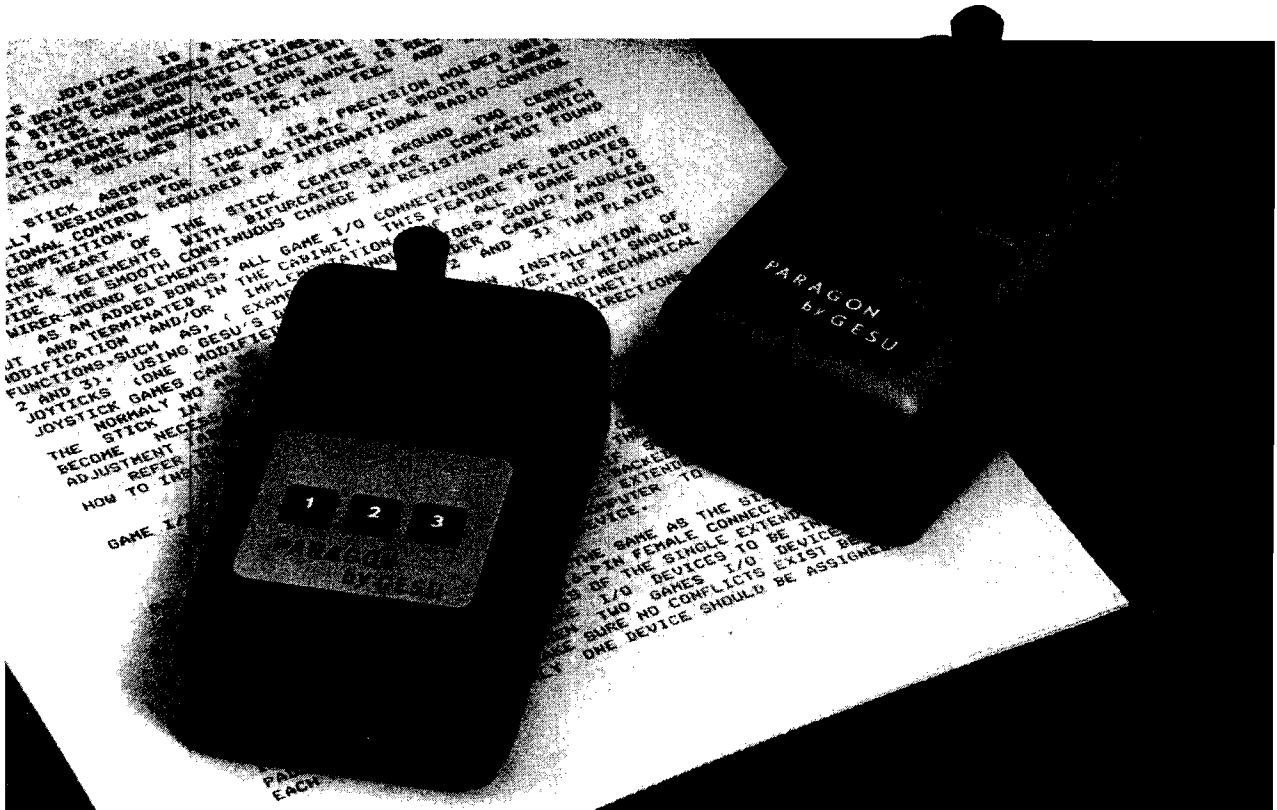
- * Building a Numeric Keypad.
- * Home Credit Card Management.
- * LO RES Shape Writing.
- * Arcade Shooting Gallery Game.
- * Random #'s in Assy. Language.
- * HI RES Weaving Design.

And many many more. NIBBLE will literally "Nibble Away" at the mysteries of the Apple II to help Beginning and Advanced Programmers, Small Businessmen, and the Whole Family enjoy and USE the Apple MORE!

It costs a paltry \$15.00 for 8 Issues! It will invite and publish user ideas and programs. DON'T WAIT! Send your check or money order right now, to receive the January issue! Mail to:

S. P. A. R. C.
P. O. Box 325
Lincoln, Mass. 01773

Software Publishing And Research Co.



ULTIMATE JOYSTICK FOR THE APPLE II

\$49.95

The **Apple Joystick** is a quality crafted dynamic interactive I/O device engineered specifically for the apple computer. The stick comes completely wired for paddles 0 & 1 and switches 0, 1 & 2. Among the excellent features of the stick are auto-centering, which positions the stick in the center of its range whenever the handle is released, and positive action switches with tactile feel and audible feedback.

The stick assembly itself is a precision molded unit originally designed for the ultimate in smooth linear proportional control required for international radio-control model competition.

The heart of the stick centers around two cermet resistive elements with bifurcated wiper contacts, which provide the smooth continuous change in resistance not found in wire-wound elements.

As an added bonus, all game I/O connections are brought out and terminated in the cabinet. This feature facilitates modification and/or implementation of all game I/O functions, such as, (example: annunciators, sound, paddles 2 and 3). Using Gesu's double I/O extender cable and two joysticks (one modified for paddles 2 and 3) two player joystick games can be implemented.

Normally no adjustment is required upon installation of the stick in your Apple computer. However, if it should become necessary to adjust the centering, mechanical adjustment tabs are provided inside the stick cabinet.

Refer to the Apple II reference manual for directions on how to install the stick in your computer.

GAME I/O EXTENDER CABLES

SINGLE \$10.00
DOUBLE \$16.00

The **single model** consists of one foot of cable, one 16-pin male and one 16-pin female connector. The extender plugs into the game I/O and the female end if secured to the outside of the cabinet with the double-backed mounting tape provided. Installed in this fashion the extender eliminates the necessity of opening the apple computer to install or remove the stick or any other game device.

The **double model** is exactly the same as the single model with the addition of a second 16-pin female connector. This extender has the same advantages as the single extender plus allowing two sticks or game I/O devices to be installed simultaneously. Note: When two games I/O devices are installed simultaneously make sure no conflicts exist between paddle assignments. Only one device should be assigned to each paddle.

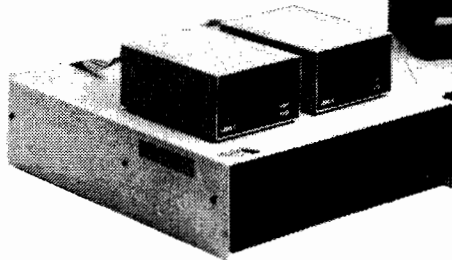
ComputerWorld

6791 WESTMINSTER AVENUE
WESTMINSTER, CA 92683
(714) 891-2587 TELEX 182274

JOIN RAYGAMCO NOW!



Become a member of
RAYGAMCO
Computer
Discount Club.



**SAVE
20%
AND MORE!**

BIG SAVINGS ON EVERY ITEM!



By being a RAYGAMCO Member you receive substantial discounts on every item you purchase, including all hardware, software, accessories, even books and paper! You will also receive a monthly newsletter with all the latest available for your particular computer system, and much, much more — exclusive to RAYGAMCO Members only!

All the famous brand names, including:

APPLE	Alpha Micro	Soroc	Lear Siegler
ATARI	Alpha Pro	Hazeltine	Shugart
EXIDY/Sorcerer	Cromemco	Sektor	Texas Instruments
Kim/Commodore	Xerox	PET	

SAVE 20% AND MORE!

Here's how to join.

Fill out the information, and mail. That's all there is to it. Nothing to buy. I want to be a RAYGAMCO Computer Discount Club Member. Please send my RAYGAMCO Membership card to:

Name _____

Address _____

City _____ State _____ Zip _____

Computer (Brand Name) _____

I would like information on (please specify system, part, accessory, book, program, etc.) _____

WE HONOR VISA, MASTERCHARGE, BANKAMERICARD.

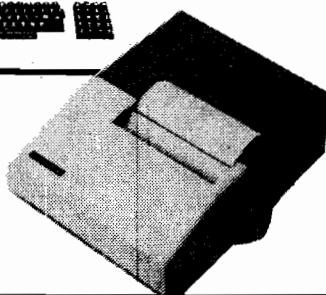
TOLL FREE, EXCEPT CA

Store Hours: Sat 10-6, Sun 12-4, Tu-Fri 11-8

800-854-6455

RAYGAM, INC.

6791 WESTMINSTER AVENUE WESTMINSTER, CA 92683
TELEX 182274 (714) 891-2587





BOX 120
ALLAMUCHY, N.J. 07820
201-362-6574

HUDSON DIGITAL ELECTRONICS INC.

THE HDE DISK SYSTEM.

HERE'S WHAT ONE USER HAS TO SAY . . .

REPRINTED BY PERMISSION FROM THE 6502 USER NOTES - ISSUE NO. 14

PRODUCT REVIEW of the HDE DISC SYSTEM by the editor.

A number of you have asked for details about the HDE full size disc system.

The system is based around the SYKES 8" drive with the 6502 based intelligent controller.

This drive is soft sector, IBM compatible, and single density which lets you store about a quarter megabyte of data on a disc.

The system software, called FODS (File Oriented Disc System), manages sequential files on the disc much the same way files are written on magnetic tape - one after another. When a file is deleted, from a sequentially managed file system, the space that the file occupied is not immediately reallocated, as in some disc operating systems. As it turns out, this can be an advantage as well as a disadvantage since deleted files on the FODS system can be recovered after the file has been deleted. (This has saved my sanity more than once!) Of course when you want to recover some of the disc space taken up by a number of these deleted files, you can simply re-pack or compress the disc and all the active files will be shifted down until there are no deleted files hanging around using up space.

FODS has this ability to repack a disc.

When saving and loading in FODS you work with named files, not track and sector data or I.D. bytes. This makes life a lot easier. I've seen some disc systems where you have to specify track and sector info and/or I.D. bytes. What a pain that can be!

If you just want to save a source file temporarily, you can do that on what's known as "scratch-pads". There are two of these on a disc, "scratch-pad A" and "scratch-pad B", each of these temporary disc files can hold up to 16K or if "B" is not used, "A" can hold one file up to 32K in length. The only files that can be temporarily saved on scratch pad are files that have been built using the system text editor.

Being a dyed in the wool assembly language programmer, I really appreciate the FODS text editor! This line oriented editor is upwards compatible with the MOS/ARESCO editor but includes about everything you could ask for in a line editor. There is a full and semi-automatic line numbering feature, lines can be edited while they are being entered or recalled and edited later, strings can be located and substituted, the line numbers can be resequenced, the file size can be found, the hex address of a line can be known and comments can be appended to an assembly file after it has been found correct. Oops! I

forgot to say lines can also be moved around and deleted. This isn't the complete list of FODS editor commands, just the ones that immediately come to mind.

Another very powerful feature of the system is the ability to actually execute a file containing a string of commands. For example, the newsletter mailing list is now being stored on disc. When I want to make labels, I would normally have to load each letter file and run the labels printing program. But with FODS, I can build up a "JOB" file of commands and execute it.

The job file in turn calls each lettered label file in and runs the label printer automatically. The way computers are supposed to operate right?

Here's a listing of the job file I use to print mailing labels:

```
:LIS PRTLBL
0005 LOD A:RUN %LABEL:LOD B:JMP.E000:
LOD C:JMP.E000:
0010 LOD D:JMP.E000:LOD E:JMP.E000:
LOD F:JMP.E000:
0015 LOD G:JMP.E000:LOD H:JMP.E000:
LOD I:JMP.E000:
0020 LOD J:JMP.E000:LOD K:JMP.E000:
LOD L:JMP.E000:
0025 LOD M:JMP.E000:LOD MC:JMP.E000:
LOD N:JMP.E000:
0030 LOD O:JMP.E000:LOD P:JMP.E000:
LOD R:JMP.E000:
0035 LOD S:JMP.E000:LOD T:JMP.E000:
LOD V:JMP.E000:
0035 LOD S:JMP.E000:LOD T:JMP.E000:
LOD V:JMP.E000:
0040 LOD W:JMP.E000:LOD XYZ:JMP.E000:
0045 LOD EXCH:JMP.E000:LOD COMP:
JMP.E000:
```

Remember the MOS/ARESCO assembler I reviewed several issues ago? Well HDE went and fixed up all the problem areas that I mentioned in the review and then took it several steps further. The HDE assembler is an honest to goodness two-pass assembler which can assemble anywhere in memory using multiple source files from the disc. The assembler is an optional part of the system.

If you're the kind of person (as I am) who enjoys having the ability to customize, modify, and expand everything you own - you'll enjoy the system expansion abilities FODS has to offer. Adding a new command is as simple as writing the program, giving it a unique three letter name and saving it to disc. Whenever you type those three letters the system will first go through its own command table, see that its not there and then go out

and read the disc directory to see if it can find it. If it's on the disc it will read it in and execute it. Simple right? I've added several commands to my system and REALLY appreciate having this ability. Some of the things I've added include a disassembler, an expanded version of XIM (the extended machine language monitor from Pyramid Data), Hypertape, and a number of system utilities which make life easier. By the way, to get back to the system, all you need to do is execute a BRK instruction.

HDE also provides a piece of software that lets you interface Microsoft 9 digit BASIC to their disc system. The software allows you to load the BASIC interpreter itself from disc as well as saving and loading BASIC Programs to and from the disc. This particular version of the software doesn't allow for saving BASIC data but HDE mentioned that this ability may be possible with a future version.

The first thing I do with a new piece of software after I get used to using it is try to blow it up. I did manage to find a weak spot or two in the very first version of FODS (a pre-release version) but the later, release version has been very tight.

The standard software that is included with the system consists of the disc driver software, the system text editor and the BASIC software interface. Several command extensions may also be included. All the necessary stuff like a power supply, the KIM-4 interface card, and all cables and connectors are included. It took me about 45 minutes to get things up and running the first time I put the system together.

Admittedly, a dual full size disc system from HDE is probably beyond the means of most hobbyists but if you or your company is looking for a dynamite 6502 development system, I would recommend this one. I've used the Rockwell System 65 while I was at MOS and feel that dollar for dollar, feature for feature, the HDE system comes out on top. The only place the HDE system falls short when stacked up next to the System 65 is in the area of packaging. At this point, there is no cabinet for the disc drives available from HDE.

So far, I've got nothing but good things to say about HDE and their products. Everything I've received from them has been industrial quality. That includes their documentation and product support. I'm very impressed with what I've seen from this company so far and quite enthusiastic over what my KIM has become since acquiring the disc system and its associated software.

ERIC

THANK YOU MR. REHNKE!

HDE PRODUCTS - BUILT TO BE USED WITH CONFIDENCE

AVAILABLE DIRECT OR FROM THESE FINE DEALERS:

JOHNSON COMPUTER
Box 523
Medina, Ohio 44256
216-725-4560

ARESCO
P.O. Box 43
Audubon, Pa. 19407
215-631-9052

PLAINSMAN MICROSYSTEMS
Box 1712
Auburn, Ala. 36830
800-633-8724

LONE STAR ELECTRONICS
Box 488
Manchaca, Texas 78652
612-282-3570

PERRY PERIPHERALS
P.O. Box 924
Miller Place, N.Y. 11764
516-744-6462

SYM-1 Tape Verification

One of the problems with using audio cassettes on any system is knowing whether or not the data has been recorded properly. By the time you find the data did not get recorded properly, it is usually too late to do anything about it. Here is a technique and program to verify the tape dump on a SYM-1.

Jack Gieryic
2041 138th Avenue, N.W.
Andover, MN 55303

Do any of you other SYMMERS ever wonder if your tape save has executed successfully? This "problem" began to haunt me more and more as my tape library grew. A fair amount of time would be lost if the data on my tape was in error. It is possible (even though remotely) two bits could be in error such that they would "cancel" each other out in the checksum verification at the end of tape read. With all this floating through my mind I decided to write the following tape verification program.

After executing a tape save (high speed format only) this program will read the data back and compare it byte for byte, to the data in the memory which you just saved. This program needs no external information (parameters) from the user. The beginning and ending addresses of the data in memory is extracted from the tape. At the end, the checksum is also verified. All the user need do is rewind the tape after a high speed format save, execute this program and then start the tape unit in the read mode.

The program is relocatable to any point in the memory. No alterations are necessary. This makes it easy to move the program into any area of memory via the MOV command. Just remember to avoid placing any part of the program near the top of page one or within the data you just saved on tape. Please note that this program is compatible with monitor version SY1.0.

```

0010:
0020:          SYM - 1 TAPE VERIFICATION
0030:          BY JACK GIERYIC
0040:          JULY, 1979
0050:
0060: 0200          ORG      $0200
0070:
0080:          MONITOR SUBROUTINES
0090:
0100: 0200          ACCESS *    $8B86
0110: 0200          CHKT   *    $8E78
0120: 0200          MONITR *    $8000
0130: 0200          OUTBYT *    $82FA
0140: 0200          RDBYTH *    $8DE2
0150: 0200          RDBYTX *    $8E28
0160: 0200          RCCHTX *    $8DDE
0170: 0200          START  *    $8DB6
0180: 0200          SYNC   *    $8D82
0190:
0200:          CONSTANTS
0210:
0220: 0200          CLKCON *    $1F
0230: 0200          SYN    *    $16
0240:
0250:          MONITOR STORAGE
0260:
0270: 0200          BUFADH *    $00FF
0280: 0200          BUFADL *    $00FE
0290: 0200          CHKH   *    $A637
0300: 0200          CHKL   *    $A636
0310: 0200          DDRIN  *    $A640
0320: 0200          DISBUF *    $A640
0330: 0200          EAH    *    $A64B
0340: 0200          EAL    *    $864A
0350: 0200          LATCHL *    $A004
0360: 0200          MODE   *    $00FD

```

Messages

If the tape agrees with the data in memory and the checksum is correct then the message "good" appears on the LED's. If the checksum is in error (even though the data compared correctly) then the message "CSUM" appears on the LED's. If any data is in error then the address of the first compare error appears on the LED's and the program terminates without checking the remainder of the data on tape.

Programming Hints

I'd like to pass along a few suggestions to you SYMMERS just getting into programming. Begin your program's (code) at location '200 (page two). Do not put anything (code, preset constants) into page one. Any constants you need in page zero should be initialized by your program. Do not set constants in page zero and then store them on tape along with your code. Do not use spare system RAM for code, constants, or temporary data storage. Begin all tape saves at location '200. Avoid saving page one on tape. I urge you to follow these suggestions as it will make your programming tasks just a bit easier.

```

0370: 0200          VIAACR *   $A00B
0380: 0200          VIAPCR *   $A00C
0390:
0400: 0200 20 86 8B BEGIN JSR ACCESS
0410: 0203 A0 80      LDYIM $80 SET MODE = HIGH SPEED
0420: 0205 20 B6 8D      JSR START INITIALIZE
0430: 0208 AD 02 A0      LDA DDRIN SET INPUT PORT
0440: 0208 29 BF          ANDIM $BF
0450: 020D 8D 02 A0      STA DDRIN
0460: 0210 A9 00          LDAIM $00
0470: 0212 8D 0B A0      STA VIAACR
0480: 0215 A9 1F          LDAIM CLKCON SET UP CLOCK
0490: 0217 8D 04 A0      STA LATCHL STORE IN LO LATCH
0500: 021A 20 82 8D LOADA JSR SYNC GET IN SYNC
0510: 021D 20 DE 8D LOADB JSR RDCHTX READ CHARACTER
0520: 0220 C9 2A          CMPIM $2A IF NOT START OF DATA
0530: 0222 F0 06          BEQ LOADC LOADC
0540: 0224 C9 16          CMPIM SYN THEN IF NOT IN SYNC
0550: 0226 D0 F2          BNE LOADA THEN RESTART SYNC SEARCH
0560: 0228 F0 F3          BEQ LOADB ELSE KEEP LOOKING FOR *
0570:                                     ELSE START OF DATA
0580:
0590: 022A A5 FD          LOADC LDA MODE CLEAR NOT IN SYNC BIT
0600: 022C 29 BF          ANDIM $BF
0610: 022E 85 FD          STA MODE
0620: 0230 20 28 8E      JSR RDBYTX READ PAST IC
0630: 0233 20 28 8E      JSR RDBYTX GET SAL FROM TAPE
0640: 0236 20 78 8E      JSR CHKT ADD TO CHECKSUM
0650: 0239 85 FE          STA BUFADL SAVE
0660: 023B 20 28 8E      JSR RDBYTX GET EAL FROM TAPE
0670: 023E 20 78 8E      JSR CHKT ADD TO CHECKSUM
0680: 0241 85 FF          STA BUFADH SAVE
0690: 0243 20 28 8E      JSR RDBYTX GET EAL FROM TAPE
0700: 0246 20 78 8E      JSR CHKT ADD TO CHECKSUM
0710: 0249 8D 4A 86      STA EAL SAVE
0720: 024C 20 28 8E      JSR RDBYTX GET EAH FROM TAPE
0730: 024F 20 78 8E      JSR CHKT ADD TO CHECKSUM
0740: 0252 8D 4B A6      STA EAH SAVE
0750:
0760: 0255 20 E2 8D LOADD JSR RDBYTH GET NEW BYTE
0770: 0258 A6 FE          LDX BUFADL IF NOT END - OF - DATA +01
0780: 025A EC 4A 86      CPX EAL
0790: 025D D0 07          BNE LOADD
0800: 025F A6 FF          LDX BUFADH
0810: 0261 EC 4B A6      CPX EAH
0820: 0264 F0 11          BEQ LOADF
0830:
0840: 0266 20 78 8E LOADD JSR CHKT THEN UPDATE CHECKSUM
0850: 0269 A0 00          LDYIM $00 IF BAD COMPARE
0860: 026B D1 FE          CMPIY BUFADL
0870: 026D D0 3D          BNE LOADG THEN ISSUE ERROR MESSAGE
0880: 026F E6 FE          INC BUFADL ELSE INC COMPARE ADDRESS
0890: 0271 D0 E2          BNE LOADD
0900: 0273 E6 FF          INC BUFADH
0910: 0275 D0 DE          BNE LOADC LOOP
0920:                                     ELSE CHECK FOR / CHARACTER
0930: 0277 C9 2F          LOADF CMPIM $2F IF NOT /
0940: 0279 D0 43          BNE LOADH THEN ERROR
0950: 027B 20 28 8E      JSR RDBYTX ELSE IF CHECKSUM IS GOOD
0960: 027E CD 36 A6      CMP CHKL
0970: 0281 D0 3B          BNE LOADH
0980: 0283 20 28 8E      JSR RDBYTX
0990: 0286 CD 37 A6      CMP CHKH
1000: 0289 D0 33          BNE LOADH
1010:                                     THEN EXIT OK
1020: 028B A2 CC          LDXIM $CC STOP TAPE
1030: 028D 8E 0C A0      STX VIAPCR
1040: 0290 A9 6F          LDAIM $6F ISSUE OK MESSAGE

```

Advertiser's Information

MICRO offers to its advertisers:

Selective readership — aimed at 6502 based computers only. Your ad does not get lost among those for other types of machines.

Effective advertising — Most of our advertisers repeat every month.

Relatively inexpensive rates — Since part of MICRO's reason for publishing is to promote the 6502, our advertising rates are kept low. In the past six months our circulation has increased 50 percent but our advertising rates have stayed the same.

Quality printing — Includes two-color advertising regularly, three- and four-color ads are available.

Regular monthly publication.

Short lead time — Approximately four weeks from advertising deadline to delivery date.

Dealer Circulation — Over half of our circulation is through stores. Your ad can be seen while customers are still deciding and buying.

Multiple Exposure — Since MICRO is kept by readers for later reference, your ad is seen over and over.

To receive our Media Kit, please contact:

MICRO
P.O. Box 6502
Chelmsford, MA 01824

```

1050: 0292 8D 41 A6      STA  DISBUF +01      "GOOD"
1060: 0295 A9 5C        LDAIM $5C
1070: 0297 8D 42 A6      STA  DISBUF +02
1080: 029A A9 5C        LDAIM $5C
1090: 029C 8D 43 A6      STA  DISBUF +03
1100: 029F A9 5E        LDAIM $5E
1110: 02A1 8D 44 A6      STA  DISBUF +04
1120: 02A4 A9 00        LDAIM $00
1130: 02A6 8D 45 A6      STA  DISBUF +05
1140: 02A9 4C 00 80      JMP  MONITR
1150:
1160: 02AC A5 FF      LOADG LDA  BUFADH DISPLAY COMPARE ERROR MESSAGE
1170: 02AE 20 FA 82      JSR  OUTBYT ADDRESS
1180: 02B1 A5 FE        LDA  BUFADL
1190: 02B3 20 FA 82      JSR  OUTBYT
1200: 02B6 A9 00        LDAIM $00
1210: 02B8 8D 41 A6      STA  DISBUF +01
1220: 02BB 4C 00 80      JMP  MONITR EXIT TO MONITOR
1230:
1240: 02BE A9 39      LOADH LDAIM $39 CHECKSUM ERROR MESSAGE
1250: 02C0 8D 42 A6      STA  DISBUF +02
1260: 02C3 A9 6D        LDAIM $6D
1270: 02C5 8D 43 A6      STA  DISBUF +03
1280: 02C8 A9 3E        LDAIM $3E
1290: 02CA 8D 44 A6      STA  DISBUF +04
1300: 02CD A9 37        LDAIM $37
1310: 02CF 8D 45 A6      STA  DISBUF +05
1320: 02D2 A9 00        LDAIM $00
1330: 02D4 8D 46 A6      STA  DISBUF +06
1340: 02D7 4C 00 80      JMP  MONITR EXIT TO MONITOR
1350:
ID=

```

Classified Ads

Superboard and C-1P Users: Sub Commander; Life; OSI 500 Race; WWI Battle (tank vs. blimp for 2); all with FULL documentation, \$8.95 each, \$27.95 for all. Ask about music systems software for live performance.

Soundsmith Software Studio
308 4th Street
Pacific Grove, CA 93950

Omni Plotting Package on disk for APPLE Computers with Applesoft on ROM. Disk and manual—\$24.00.

Axe Software International
237 Star Rte.

Santa Barbara, CA 93105

BASIC program manual has 96 pages of step-by-step instructions for OSI Challenger 1P and Superboard II owners. Order "Getting Started with Your C1P" from

TIS

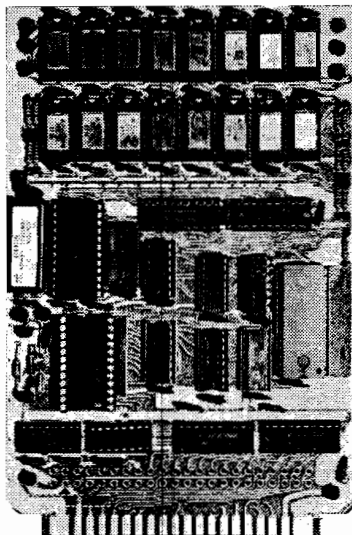
Box 921, Dept. M
Los Alamos, NM 87544

Price: \$5.95 plus \$1 postage and handling

ADVERTISE IN MICRO

A classified ad may be run for \$10.00 per month. Ad should not exceed 6 lines, one per person (or company). Must relate to 6502 industry, must be prepaid. Ad deadline: 25th of month.

KIM/SYM/AIM-65—32K EXPANDABLE RAM DYNAMIC RAM WITH ON BOARD TRANSPARENT REFRESH THAT IS COMPATIBLE WITH KIM/SYM/AIM-65 AND OTHER 6502 BASED MICROCOMPUTERS.



ASSEMBLED/ TESTED	WITH 32K RAM	\$419.00
	WITH 16K RAM	\$349.00
	WITHOUT RAM CHIPS	\$279.00
	HARD TO GET PARTS ONLY (NO RAM CHIPS) BARE BOARD AND MANUAL	\$109.00 \$49.00

- * PLUG COMPATIBLE WITH KIM/SYM/AIM-65. MAY BE CONNECTED TO PET USING ADAPTOR CABLE. SS44-E BUS EDGE CONNECTOR.
- * USES +5V ONLY (SUPPLIED FROM HOST COMPUTER BUS). 4 WATTS MAXIMUM.
- * BOARD ADDRESSABLE IN 4K BYTE BLOCKS WHICH CAN BE INDEPENDENTLY PLACED ON 4K BYTE BOUNDARIES ANYWHERE IN A 64K BYTE ADDRESS SPACE.
- * ASSEMBLED AND TESTED BOARDS ARE GUARANTEED FOR ONE YEAR, AND PURCHASE PRICE IS FULLY REFUNDABLE IF BOARD IS RETURNED UNDAMAGED WITHIN 14 DAYS.
- * BUS BUFFERED WITH 1 LS TTL LOAD.
- * 200NSEC 4116 RAMS.
- * FULL DOCUMENTATION

PET INTERFACE KIT \$49.00

CONNECTS THE ABOVE 32K EXPANDABLE RAM TO A 4K OR 8K PET.
CONTAINS EXPANSION INTERFACE CABLE, BOARD STANDOFFS,
POWER SUPPLY MODIFICATION KIT AND COMPLETE INSTRUCTIONS.

6502, 64K BYTE RAM AND CONTROLLER SET
MAKE 64K BYTE MEMORY FOR YOUR 6800 OR 6502. THIS CHIP SET INCLUDES:
* 32 MSK 4116-3 16KX1, 200 NSEC RAMS.
* 1 MC3480 MEMORY CONTROLLER.
* 1 MC3242A MEMORY ADDRESS MULTIPLEXER AND COUNTER.
* DATA AND APPLICATION SHEETS. PARTS TESTED AND GUARANTEED.
\$325.00 PER SET

16K X 1 DYNAMIC RAM
THE MK4116-3 IS A 16,384 BIT HIGH SPEED NMOS DYNAMIC RAM. THEY ARE EQUIVALENT TO THE MOSTEK, TEXAS INSTRUMENTS, OR MOTOROLA 4116-3
* 200 NSEC ACCESS TIME, 375 NSEC CYCLE TIME
* 16 PIN TTL COMPATIBLE
* BURNED IN AND FULLY TESTED
* PARTS REPLACEMENT GUARANTEED FOR ONE YEAR
\$8.50 EACH IN QUANTITIES OF 8

BETA COMPUTER DEVICES
P.O. BOX 3465
ORANGE, CALIFORNIA 92665
(714) 633-7280

CALIF. RESIDENTS PLEASE ADD 4% SALES TAX
MASTERCHARGE & VISA ACCEPTED. PLEASE
ALLOW 14 DAYS FOR CHECKS TO CLEAR BANK.
PHONE ORDERS WELCOME

ALL ASSEMBLED BOARDS AND MEM-
ORY CHIPS CARRY A FULL ONE YEAR
REPLACEMENT WARRANTY.

Announcing...

OPTIMIZED SYSTEMS SOFTWARE
UPGRADE YOUR APPLE II[®] WITH A NEW SYSTEMS SOFTWARE PACKAGE

- Unified Operating System
- Disk File Manager
- Commercial Basic
- Editor/Assembler/Debugger
- Data Base Manager

Optimized Systems Software does not use Apple DOS[®]. OSS is a unified and complete systems software package with its own Operating System and File Manager. The Operating System, the File Manager and the Basic combined use only slightly more RAM than Apple DOS[®] alone. Requires 48K Apple II[®] with Disk II.

Operating System

- Byte and Block I/O
- Simple User Interface
- Simple Device Interface
(create your own)

Basic

- Nine Digit Precision DECIMAL Floating Point
- 32K Byte Strings
- Variable Names to 256 significant characters
- I/O Interface Statements
(no PRINT "control-D...")

File Manager

- Open, Read, Write, Delete, Lock, etc.
- Random Access via Note & Point
- File Names of Primary.Ext type

Editor/Assembler/Debugger

- Line Editor
(Edits Basic programs, too)
- Mini Assembler
- Maxi Assembler
- Disassembler
- Step, Trace, etc.

Available NOW at Special Introductory Prices

- | | |
|---|---------|
| ● Operating System + File Manager | \$24.95 |
| ● Operating System + File Manager + Basic | \$49.95 |
| ● Operating System + File Manager + ASM | \$49.95 |
| ● Operating System + File Manager + Basic + ASM | \$89.95 |
| ● Operating System + Data Base Manager | (2nd Q) |

Order today. Add \$2.00 for shipping & handling. California residents add 6% sales tax. Visa/Mastercharge welcome. Personal checks require 2 weeks to clear.

Note: Apple II[®], Apple DOS[®] are trademarks of Apple Computer, Inc.

Optimized Systems Software
Shepardson Microsystems, Inc.
20823 Stevens Creek Blvd., Bldg. C4-H
Cupertino, CA 95014
(408) 257-9900

Microbes and Miscellanea

R. M. Mottola from Boston, MA writes:

It has been brought to my attention that my Screen Dump Software (14:27) will not work with a printer that can handle more than 40 columns. To correct this, please make the following changes:

```
580 NEXT:PRINT "" :REM Null $
```

```
585 NEXT:NEXT
```

These changes will provide the carriage return that 40 column printers add automatically.

I'd also like to thank William Luebbert for his APPLE II memory map. It is the most valuable article I've read in a long time.

Jack Gieryc of Andover, MN found a disturbing bit of information in the July 1979 issue of MICRO, Nicholas Vritis' article "The First Book of KIM—on a SYM":

Mr. Vrtis recommended a hardware modification to remove the jumper enabling system RAM write protect, jumper MM-45. His alternative to this modification is to insert a JSR ACCESS in order to remove the write protect.

I strongly urge all SYM owners to use the JSR ACCESS to free up system RAM prior to code which writes into system RAM and, if possible, the JSR NACCES after your code to once again write protect system RAM. Do not remove jumper MM-45.

I have two reasons for urging avoidance of the hardware change. First, your program may contain a bug elsewhere which inadvertently writes into some or all of system RAM. Permanently removing the write protect feature will make this bug more difficult to trace. Instead of "missing data" in some buffer or variable (a problem relatively easy to "see" and figure out) you may have memory alterations which could be impossible to view as a critical element of system RAM was destroyed.

The second reason looks to the future a bit. If Synertek ever does add a disk option to the SYM, I wouldn't be surprised if critical information relating to the disk driver were located in the system RAM. If so, a bug which alters this memory could also cause your disk data to be destroyed. This supposition does assume quite a bit but is not outside the realm of possibility.

Philip L. Bryan suggests that in the article of Robert Carlson's, "Baudot Teletype Driver", in the Sept. issue, the op-code for RORA should be 6A, not 68.

Robert A. Peck of Sunnyvale, CA says:

I tried the SYMphony in Stereo program in June 1979 MICRO, and ran into some difficulties which I have fixed for my machine, and I wanted to let you know about the problems.

Problem A: Program goes from 0200-0278, data area overlaps—0270-03F2. *Actions:* Begin data area at 279, GO TO 3FB change 0004 to 79, 0005 to 38.

Problem B: Data for the starting tune addresses is picked up from the wrong locations. At 0219 and 0223, the instruction "B1 04" is used. This will pick up the data bytes stored in locations 4 & 5, add the Y register contents, and use this as the effective address of the data to be loaded into location 0, then location 1. After execution, location zero contains "05" and location one will contain the contents of location 2F02. This combination XX05 does not match the starting address of the note table. *Actions:* Change 021A to "1D", change 0224 to "10", store "04" at 0010, and store "00" at 0011.

Problem C: Second half of tune (part 2 of table) never starts, always stays within first part. *Action:* Change instruction at 0239 to read:
0239 30 37 BMI 0272 GO TO BUMP.

Problem D: On completion of the tune, it goes back and repeats the second half only—on completion of a tune, any repeat should repeat the ENTIRE tune. *Action:* Change 0230 to "E2".

It works fine with these changes.

George Shirn of Williamstown, MA has this idea:

If you have updated your SYM—1 with the new monitor, MON 1.1, then John Gieryc's SYM—1 Tape Directory, (8:35) needs changing.

Change	From	To
0206	B6	A9
021B	82	52
021E	DE	E1
0231	28	26
0236	28	26
023B	28	26
0204	E2	E5
0245	E2	E5
02BC	0B	06

Then it works fine.

More MICROBES on following page...

From LeRoy Moyer of Charlottesville, VA:

MICRO contains many articles which I enjoy, particularly those that deal with machine language routines. In the November issue the Applesoft Renumbering program was a very useful addition. One modification that I made to the program that other readers may be interested in is to include a

CMP #BC
BEQ \$6D20

This is put in the vicinity of \$6CF7 to 6D01 and will then also do the LIST function.

Bob Bishop of Mountain View, CA has corrections for his article, "APPLE II Hires Picture Compression" (18:17):

On page 23, under listing 2, there should be at 0CC0:

A0, 00, 84, 03,
A2, 40, 86, 04,
98, 91, 03, C8,
D0, FB, E6, 04,
CA, D0, F6, 60.

Also in Listing 3 at 1280 Hexidecimal it should read:

8A, 48, 98, 48,
A5, 10, 8D, C1
83, A9, C2, 85,
OE, A9, 83, 85,
OF, A9, 00, 8D,
CO, 83, 85, 00,
85, 01, A5, 01,
4A, 09, 60, 85,

OD, A9, 00, 6A,
65, 00, 85, 0C,
A2, 08, A0, 00,
B1, 0C, 91, OE,
E6, OE, D0, 02,
E6, OF, 18, A5,
OD, 69, 04, 85,
OD, CA, D0, EC,
EE, CO, 83, A5,
00, C5, 07, D0,
06, A5, 01, C5,
08, FO, 10, E6,
00, A5, 00, C9,
28, D0, C3, A9,
00, 85, 00, E6,
01, D0, BB, 68,
A8, 68, AA, 60.

E.D. Morris of Midland, MI informs us that:

The article "Tokens" which appeared in the August issue of MICRO was actually co-authored by myself and Al Adams, 407 Rollcrest, Midland, MI 48640.

And Elsa Lewis from Chapel Hill, NC wonders about our illustrations...

The article on writing for MICRO (17:59) had some practical ideas. However, the one thing I found out of line was the etches! Maybe the person depicted should have been using a microcomputer, printer, and word processor to compose the article. Would anybody owning a micro still be plunking away at a typewriter?

T.D.Q.

TAPE DATA QUERY

THE IDEAL SOLUTION FOR PERSONAL AND VERY-SMALL BUSINESS DATA MANAGEMENT

PET-8K

TRS-80-LVL II

- * COMPLETE CASSETTE FILE MANAGEMENT SYSTEM
 - ENGLISH-LIKE COMMAND LANGUAGE
 - REPORT GENERATOR
 - UTILITY PACKAGE
 - NO PROGRAMMING KNOWLEDGE REQUIRED
 - REQUIRES 2 CASSETTE RECORDERS
- * T.D.Q. APPLICATION CASEBOOK

— COMPLETE DIRECTIONS TO MICRO-COMPUTERIZE:

- INVENTORY CONTROL
- ACCOUNTS RECEIVABLE
- ORDER PROCESSING
- LABEL PRINTING
- CHECK PRINTING
- INVOICE PRINTING
- CUSTOMER DIRECTORY
- APPOINTMENT SCHEDULING
- VENDOR MASTER FILE
- PAYROLL JOURNAL
- CHECKBOOK JOURNAL
- TELEPHONE BOOK
- RENT COLLECTION

** SPECIAL YEAR-END SALE PRICE — \$100.00** — INCLUDES: CASEBOOK; 2 CASSETTES; 3 USER'S MANUALS & REF. CARDS

ORDERS MUST BE RECEIVED BY JAN. 31, 1980
SEND CHECK OR MONEY-ORDER TO:

H. GELLER COMPUTER SYSTEMS

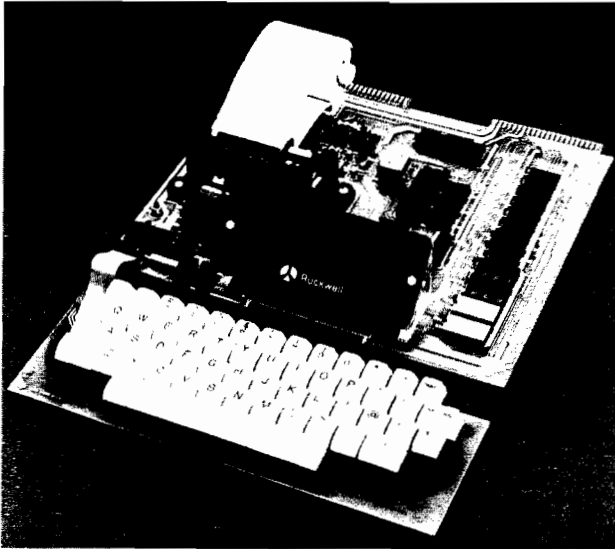
DEPT. M

P.O. BOX 350

NEW YORK, N.Y. 10040

(N.Y. RESIDENTS ADD SALES TAX)

AIM 65 BY ROCKWELL INTERNATIONAL



AIM 65 is fully assembled, tested and warranted. With the addition of a low cost, readily available power supply, it's ready to start working for you.

AIM 65 features on-board thermal printer and alphanumeric display, and a terminal-style keyboard. It has an addressing capability up to 65K bytes, and comes with a user-dedicated 1K or 4K RAM. Two installed 4K ROMs hold a powerful Advanced Interface Monitor program, and three spare sockets are included to expand on-board ROM or PROM up to 20K bytes.

An Application Connector provides for attaching a TTY and one or two audio cassette recorders, and gives external access to the user-dedicated general purpose I/O lines.

Also included as standard are a comprehensive AIM 65 User's Manual, a handy pocket reference card, an R6500 Hardware Manual, an R6500 Programming Manual and an AIM 65 schematic.

AIM 65 is packaged on two compact modules. The circuit module is 12 inches wide and 10 inches long, the keyboard module is 12 inches wide and 4 inches long. They are connected by a detachable cable.

THERMAL PRINTER

Most desired feature on low-cost microcomputer systems . . .

- Wide 20-column printout
- Versatile 5 x 7 dot matrix format
- Complete 64-character ASCII alphanumeric format
- Fast 120 lines per minute
- Quite thermal operation
- Proven reliability

FULL-SIZE ALPHANUMERIC KEYBOARD

Provides compatibility with system terminals . . .

- Standard 54 key, terminal-style layout
- 26 alphabetic characters
- 10 numeric characters
- 22 special characters
- 9 control functions
- 3 user-defined functions

TRUE ALPHANUMERIC DISPLAY

Provides legible and lengthy display . . .

- 20 characters wide
- 16-segment characters
- High contrast monolithic characters
- Complete 64-character ASCII alphanumeric format

PROVEN R6500 MICROCOMPUTER SYSTEM DEVICES

Reliable, high performance NMOS technology . . .

- R6502 Central Processing Unit (CPU), operating at 1 MHz. Has 65K address capability, 13 addressing modes and true index capability. Simple but powerful 56 instructions.
- Read/Write Memory, using R2114 Static RAM devices. Available in 1K byte and 4K byte versions.
- 8K Monitor Program Memory, using R2332 Static ROM devices. Has sockets to accept additional 2332 ROM or 2532 PROM devices, to expand on-board Program memory up to 20K bytes.
- R6532 RAM-Input/Output-Timer (RIOT) combination device. Multipurpose circuit for AIM 65 Monitor functions.
- Two R6522 Versatile Interface Adapter (VIA) devices, which support AIM 65 and user functions. Each VIA has two parallel and one serial 8-bit, bidirectional I/O ports, two 2-bit peripheral handshake control lines and two fully-programmable 16-bit interval timer/event counters.

BUILT-IN EXPANSION CAPABILITY

- 44-Pin Application Connector for peripheral add-ons
- 44-Pin Expansion Connector has full system bus
- Both connectors are KIM-1 compatible

TTY AND AUDIO CASSETTE INTERFACES

Standard interface to low-cost peripherals . . .

- 20 ma. current loop TTY interface
- Interface for two audio cassette recorders
- Two audio cassette formats: ASCII KIM-1 compatible and binary, blocked file assembler compatible

ROM RESIDENT ADVANCED INTERACTIVE MONITOR

Advanced features found only on larger systems . . .

- Monitor-generated prompts
- Single keystroke commands
- Address independent data entry
- Debug aids
- Error messages
- Option and user interface linkage

ADVANCED INTERACTIVE MONITOR COMMANDS

- Major Function Entry
- Instruction Entry and Disassembly
- Display/Alter Registers and Memory
- Manipulate Breakpoints
- Control Instruction/Trace
- Control Peripheral Devices
- Call User-Defined Functions
- Comprehensive Text Editor

LOW COST PLUG-IN ROM OPTIONS

- 4K Assembler—symbolic, two-pass \$79.00
- 8K BASIC Interpreter \$99.00

POWER SUPPLY SPECIFICATIONS

- +5 VDC \pm 5% regulated @ 2.0 amps (max)
- +24 VDC \pm 15% unregulated @ 2.5 amps (peak)
0.5 amps average

PRICE: \$369.00 (1K RAM)

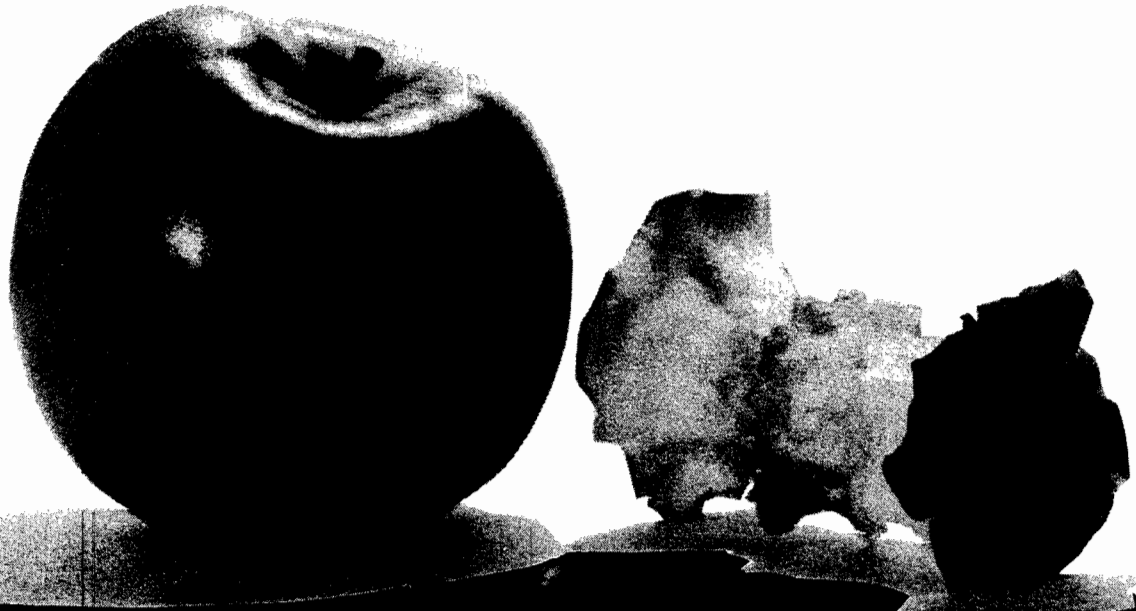
Plus \$4.00 UPS (shipped in U.S. must give **street** address), \$10 parcel post to APO's, FPO's, Alaska, Hawaii, Canada, \$25 air mail to all other countries

We manufacture a complete line of high quality expansion boards. Use reader service card to be added to our mailing list, or U.S. residents send \$1.00 (International send \$3.00 U.S.) for airmail delivery of our complete catalog.

 **ENTERPRISES**
INCORPORATED

2967 W. Fairmount Avenue
Phoenix AZ 85017
(602)265-7564





Introducing AppleSeed, our newest publication to whet your Apple* appetite!

We invite you to subscribe to AppleSeed - the magazine that is to the Apple II* what SoftSide is to the TRS-80**. It offers the newest in software programming hints and ideas tailored especially for your computer. AppleSeed features challenging programs for both the do-it-yourselfer and the individual interested in pre-packaged programs and games . . . your own preview of the best available on the market today. A typical slice of AppleSeed consists of one major (new 16K) commercial level program (completely listed for your keying pleasure), accompanied by two or three applications for practical use or fun, supplemented by informative articles to polish your Apple*. Get right to the core of your Apple* needs and order AppleSeed today! 12 issues, 1 year, \$15.00. AppleSeed is the newest member of . . .

SoftSide™
PUBLICATIONS

6 South Street, Millford, NH 03055
(803) 673-5144

*A registered trademark of Apple Computers. **A registered trademark of Radio Shack and Tandy Corp.

Symbol Table Sorter/Printer for the AIM Assembler

Some information about the AIM Assembler, a program to print the Symbol Table - sorted alphabetically or numerically, and some other useful stuff.

Mel Evans
ERIM, P.O. Box 8618
Ann Arbor, MI 48107

When the first Rockwell AIM showed up at the local computer store, mouths started watering. For a KIM user, to see an AIM is to want one. It is hard to resist that fine keyboard and display, the clever little printer, and sockets for Monitor, RAM, Assembler, and BASIC; or for 2716 EPROM with your own stuff on it. I've been running KIM with a Memory Plus board (8K RAM, 8K EPROM, 2716 programmer, and a 6522 VIA), mounted with power supply and I/O board in an attache case for portable use. This rig has accumulated a half-dozen 2716's full of KIM software, and I intend to continue working on KIM applications. Since AIM provides the same VIA, I bought one with the justification that it would help me develop more and better KIM software. If you write it and debug it on AIM, and move it over to KIM, you're done, right?

Well, yes. After a bit of learning about conversion from one memory map to another, it really does work that way. The mnemonic insert mode ("I" command) is a joy to use. There are no more op-code lookups and branch calculations and there are fewer typos. And the disassembler ("K" command) lets you check your work faster and more accurately. But for clean, patch-free object code, the assembler is the best of all. Six-character variable names! No line-number hassle! Six-character labels, such as "JMP NEXT," or "BEQ OUTCHR." And for easy transfer of object code from AIM to KIM, it's the assembler that really does it. It makes the writing of relocatable code almost automatic.

The AIM assembler lacks one feature; there is no command for printing the symbol table after an assembly. So here is a little program that fits on Page Zero and does just that. After assembling any program, load this one and start at 10. It prints two listings of the assembly symbol table; one sorted alphabetically by symbol name, and the other sorted numerically by symbol address. The first list is helpful when going through the assembly listing. The second is even more helpful when reading the output of the disassembler; it lets you know right away that the cryptic "JSR E9BC," for example, is a jump to subroutine OUTALL.

The source (assembly-language) version of the sort/print program is shown in Figure 1. The assembly listing, with absolute addresses, is shown in Figure 2. A disassembler listing is not shown; if you can't assemble this one, you don't need it!

The sorting algorithm is plain brute-force; it is designed to conserve memory space, not sorting time. But even so, it takes much less time to sort a list than it does to print it. The only tricky feature of the program is in its allocation of zero-page memory; in loading, it carefully avoids wiping out the six bytes that remember symbol-table size and location, because it will need them to know where to work when you hit "Go."

Figure 3 shows, as an example, the use of the assembled program on its

own symbol table. Notice that you don't have to find and enter the location and size of the symbol table; the program finds these from the zero-page bytes that it conserved while loading.

One note of caution in case you don't read the following section. When you assemble this source program, don't direct the object code to memory. Direct it to tape. Then load it and start at 10.

AIM-to-KIM Software Conversion

The following assumes that you have more space in AIM RAM than you will need for KIM memory. It works well with a 4K AIM, and even better with 8K.

The idea is to use AIM for both assembly and running of the program during the debug phase. In the process of editing source, assembling, and running (and re-editing, re-assembling, re-running, re-editing, etc., etc.), much time can be saved by not having to load source from tape, dump object to tape, and reload object from tape for the next run. (If you have disc, this may be less of a problem. I wouldn't know.) So, build your source with the editor (the very good editor), assemble from memory, and direct object to memory — to any available memory, not necessarily where it will go in KIM. It will be easy to move later if you follow one rule: don't use fixed addresses except where really necessary.

Look at Figure 1 again. Observe that the only fixed addresses used are those

of the six zero-page bytes containing symbol-table location and size (STLO through NSYMH1), the four Monitor subroutines needed for printing (CLR through CRCK), the start of the scratchpad block ($\star = \$00$), and the start of the main program ($\star = \$10$). All other addressing is either relative ($\star = \star + 1, \star = \star + 4$) or by label (JSR SORT, JMP COMPAR, BNE SWAP), with absolute addresses and branch offsets assigned during assembly. Therefore, this whole program could be moved to KIM by simply changing the scratchpad start to any convenient spot in KIM zero-page, changing the program start to any appropriate spot in KIM RAM, and re-assembling, with object-output to tape in KIM format.

That last phrase, "output to tape in KIM format," is where we hit the first snag. The AIM User's Manual says the assembler will do this, but the manual is wrong. If you try OUT-OBJ = K, the poor thing locks up in a trance, and the only recovery is RESET. (If you would like an explanation from Rockwell on why this happens, call Dave Sawtelle, AIM Applications, 714-632-0975. This number is worth writing down; AIM Applications is a very competent and helpful group.)

So how do you output object to tape in KIM format? You have your choice of two ways. The simple way is to output object to tape in AIM format, load this back into AIM, and then DUMP it to tape in KIM format. This works fine, but it is slow. The faster way, if you have room in AIM RAM, is to send object to memory and then DUMP in KIM format. Before you do either, read on, or you may hit the second snag.

The above sort/print is a bad example of KIM-convertible code, for two reasons. The first is obvious; considering its function, KIM couldn't do anything with it. The second illustrates some further precautions.

The AIM editor and assembler use the top third (and some of the bottom) of Page Zero, and several pieces of Page One are used by tape I/O and monitor. Furthermore, you can't (yet) trust the memory map, in the User's Manual. Rockwell is diligently fixing the mistakes and has already issued Revision 1, but it is still too new to be totally reliable. For example, look at the equate list in Fig. 1 again. Notice those zero-page addresses for STLO through NSYMH1? Does the memory map tell you they are used by the assembler? No, it doesn't. STLO, STHI, NSYMLO, and NSYMH1 are mentioned in the chapter on the assembler (Section 5.2). I found ENLO and ENHI by accident!

In order to assemble to memory and run, try to avoid putting either program or data on either Page Zero or Page One,

unless you want to discover, by trial and error, the undocumented portions of the memory map. It's okay to assign zero-page variables, but don't use the assembler to initialize them with data. The data may not survive the assembly.

Now, how about a program destined for Page Zero, such as the sorter/printer above? The final version (as listed above) must be assembled with object-output to tape, and can then be safely loaded and run. But during debug, the assemble-to-memory-and-run cycle can still be used by moving program and data to higher memory. For example, just before YTAB, change " $\star = \star + 1$ " to " $\star = \$200$ " (to move data to Page 2); and before START, change " $\star = \$10$ " to " $\star = \$300$ " (to move the program to Page 3). This changes some addressing modes from zero-page to absolute, but the assembler takes it in stride. Now assemble to memory and run. After it all works, move data and program down to Page Zero, and assemble to tape.

What if you need to use Page One? The push-down stack at the top of Page One is the same in AIM as in KIM, so there is no problem there. (Simply allow a bit more room for the deeper-pushing AIM monitor.) The AIM memory map shows eleven Page One bytes (106-107, 115-11D) used by tape I/O, and eight bytes (168-16F) used by the monitor. The tape I/O bytes can be handled like Page Zero bytes; i.e., avoid until assembling to tape. The eight monitor bytes should probably be permanently avoided; load them into KIM by hand after everything else is transferred. And as an extra precaution, check all of Page One for wipeouts before running in KIM.

Please do not let all these cautions scare you off. It really is fast and easy after a little practice. Most programs grow during debug, and much of the above only applies if your program has grown to the point where you are cramped for memory space.

Fig. 4 shows how simple it is when there is plenty of room. This is a general-purpose "move block" program that will go anywhere in memory (RAM or ROM), and it will move any size block from anywhere to anywhere. The assembly listing (Fig. 5) shows that it occupies 24 HEX bytes of memory, and uses six bytes of zero-page. Before moving it to KIM, change that " $\star = \$00$ " to the start of the six-byte block you want it to use in KIM. Don't bother to change the " $\star = \$200$ " starting address; after you have it in KIM, you can use the program to move itself to wherever you want to keep it. I keep two copies on tape, one that loads to zero-page and one to the top of RAM, plus one more in EPROM. With another copy in AIM, it can be used for general memory transfer in either

direction; move blocks to \$200-3FF, dump to tape, load to \$200-3FF in the other machine, and move to wherever.

If all you want is the block-move code, Fig. 6 gives a disassembler listing and a hex dump. It can be put anywhere, but this version needs the bottom six bytes of zero-page for "From", "End", and "To".

Figure 1: Source Listing, Sorter/Printer

```

;SPSYM
;
;SORT & PRINT SYMBOL
;  TABLE AFTER
;  AIM ASSEMBLY
;
;EQUATE LIST
;
STLO=$3A
STHI=$3B
ENLO=$3C
ENHI=$3D
NSYMLO=$0C
NSYMH1=$0B
;
CLR=$EB44
OUTALL=$E9BC
NUMA=$EA46
CRCK=$EA24
;
;
*=$00
CNTLO
*=$+1
CNTHI
*=$+1
ADLO
*=$+1
ADHI
*=$+1
Y1
*=$+1
Y2
*=$+1
YLIM
*=$+1
YTAB
;
; YTAB DATA
; DBY $0008, $060E
;
; MAIN PROGRAM
; JUMP OVER 0B, 0C

```

```

) (TO $10)
*=$10
START
; SORT BY NAME
LDA #6
STA YLIM
LDX #0
JSR SORT
; SORT BY ADDRESS
LDA #8
STA YLIM
LDX #2
JSR SORT
BRK
BRK
)
; SUBROUTINES
)
SORT LDA NSYMLO
STA CNTLO
LDA NSYMHI
STA CNTHI
SRT1 JSR SETADR
SRT2 LDA YTAB,X
STA Y1
LDA YTAB+1,X
STA Y2
; JUMP OVER 3A-3D
JMP COMPAR
**++4
; COMPARE CHAR. W/
; CORRESP. CHAR. IN
; NEXT LINE.
; IF A<B, NEXT LINE
; IF A>B, SWAP.
; IF A=B, NEXT CHAR.
COMPAR LDY Y1
LDA (ADLO),Y
LDY Y2
CMP (ADLO),Y
BCC NXLINE
BNE SWAP
INC Y1
INC Y2
LDA Y1
CMP YLIM
BNE COMPAR
SWAP LDA #0
STA Y1
LDA #8
STA Y2
SWP1 LDY Y1
LDA (ADLO),Y
PHA
LDY Y2

```

```

LDA (ADLO),Y
LDY Y1
STA (ADLO),Y
PLA
LDY Y2
STA (ADLO),Y
INC Y1
INC Y2
LDA Y1
CMP #8
BNE SWP1
NXLINE JSR INCADR
BNE SRT2
; DECREMENT
; LOOP COUNT
SEC
LDA CNTLO
SBC #1
STA CNTLO
BCS **++4
DEC CNTHI
LDA CNTHI
BNE **++4
LDA CNTLO
BNE SRT1
; PRINT SORTED LIST
JSR SETADR
JSR GAP
PRNT1 JSR CLR
LDY #4
PR1A LDA ##20
JSR OUTALL
DEY
BNE PR1A
LDY #0
PRNT2 LDA (ADLO),Y
JSR OUTALL
INY
CPY #6
BNE PRNT2
LDA ##20
JSR OUTALL
PRNT3 LDA (ADLO),Y
JSR NUMA
INY
CPY #8
BNE PRNT3
JSR CRCK
JSR INCADR
BMI PRNT1
BEQ PRNT1
JSR GAP
RTS
)
SETADR LDA STLO

```

```

STA ADLO
LDA STHI
STA ADHI
RTS
INCADR CLC
LDA ADLO
ADC #8
STA ADLO
BCC **++4
INC ADHI
LDA ADHI
CMP ENHI
BNE INAX
LDA ADLO
CMP ENLO
INAX RTS
)
GAP LDX #3
GP1 JSR CLR
LDA ##20
JSR OUTALL
JSR CRCK
DEX
BNE GP1
LAST RTS
)
.END
)
; SPSYM
)
; SORT & PRINT SYMBOL
; TABLE AFTER
; AIM ASSEMBLY
)
; EQUATE LIST
)
==0000 STLO=$3A
==0000 STHI=$3B
==0000 ENLO=$3C
==0000 ENHI=$3D
==0000 NSYMLO=$0C
==0000 NSYMHI=$0B
)
==0000 CLR=$EB44
)

```

Fig. 2: Assembly Listing, Sorter/Printer

```

==0000 OUTALL=$E9BC      ; SUBROUTINES
                          ;
==0000 NUMA=$EA46       ==0024 SORT
                          A500 LDA NSYMLO
                          8500 STA CNTLO
                          A508 LDA NSYMHI
                          8501 STA CNTHI
                          ;
==0000                  ==002C SRT1
                          20CC00 JSR SETADR
                          ==002F SRT2
                          B507 LDA YTAB,X
                          8504 STA Y1
                          B508 LDA YTAB+1,X
                          8505 STA Y2
                          ; JUMP OVER 3A-3D
                          4C3E00 JMP COMPAR
                          ==003A
                          ;
                          ;*****4
                          ; COMPARE CHAR. W/
                          ; CORRESP. CHAR. IN
                          ; NEXT LINE.
                          ; IF A<B, NEXT LINE
                          ; IF A>B, SWAP.
                          ; IF A=B, NEXT CHAR.
                          ==003E COMPAR
                          A404 LDY Y1
                          B102 LDA (ADLO),Y
                          A405 LDY Y2
                          D102 CMP (ADLO),Y
                          9030 BCC NXLINE
                          D00A BNE SWAP
                          E604 INC Y1
                          E605 INC Y2
                          ==004E
                          A504 LDA Y1
                          C506 CMP YLIM
                          D0EA BNE COMPAR
                          ==0054 SWAP
                          A900 LDA #0
                          8504 STA Y1
                          A908 LDA #8
                          8505 STA Y2
                          ==005C SWP1
                          A404 LDY Y1
                          B102 LDA (ADLO),Y
                          48 PHA
                          A405 LDY Y2
                          B102 LDA (ADLO),Y
                          A404 LDY Y1
                          9102 STA (ADLO),Y
                          68 PLA
                          A405 LDY Y2
                          ==006C
                          9102 STA (ADLO),Y
                          E604 INC Y1
                          E605 INC Y2
                          A504 LDA Y1
                          C908 CMP #8
                          D0E4 BNE SWP1
                          ==0078 NXLINE
                          20D500 JSR INCADR
                          D0B2 BNE SRT2
                          ; DECREMENT
                          ; LOOP COUNT
                          38 SEC
                          A500 LDA CNTLO
                          E901 SBC #1
                          8500 STA CNTLO
                          B002 BCS *+4
                          C601 DEC CNTHI
                          ==0088
                          A501 LDA CNTHI
                          D002 BNE *+4
                          A500 LDA CNTLO
                          D09C BNE SRT1
                          ; PRINT SORTED LIST
                          20CC00 JSR SETADR
                          20EB00 JSR GAP
                          ==0096 PRNT1
                          2044EB JSR CLR
                          A004 LDY #4
                          ==009B PR1A
                          A920 LDA #$20
                          20BCE9 JSR OUTALL
                          08 DEY
                          D0F8 BNE PR1A
                          A000 LDY #0
                          ==00A5 PRNT2
                          B102 LDA (ADLO),Y
                          20BCE9 JSR OUTALL
                          C8 INY
                          C006 CPY #6
                          D0F6 BNE PRNT2
                          A920 LDA #$20
                          20BCE9 JSR OUTALL
                          ==00B4 PRNT3
                          B102 LDA (ADLO),Y
                          2046EA JSR NUMA
                          C8 INY
                          C008 CPY #8
                          D0F6 BNE PRNT3
                          2024EA JSR CRCK
                          20D500 JSR INCADR
                          ==00C4
                          30D0 BMI PRNT1
                          F0CE BEQ PRNT1
                          20EB00 JSR GAP
                          60 RTS
                          ;
                          ==00CC SETADR

```

```

A53A LDA STLO
8502 STA ADLO
A538 LDA STHI
8503 STA ADHI
60 RTS
;
==00D5 INCADR
18 CLC
A502 LDA ADLO
6908 ADC #8
8502 STA ADLO
9002 BCC **+4
E603 INC ADHI
A503 LDA ADHI
C53D CMP ENHI
D004 BNE INAX
==00E6
A502 LDA ADLO
C53C CMP ENLO
==00EA INAX
60 RTS
;
==00EB GAP
A203 LDX #3
==00ED GP1
2044EB JSR CLR
A920 LDA ##20
20BCE9 JSR OUTALL
2024EA JSR CRCK
CA DEX
D0F2 BNE GP1
==00FB LAST
60 RTS
;
END
ERRORS= 0000

```

Fig. 3: Example Run showing Dual Sort

```

<*>=10
<G>/

```

```

ADHI 0003
ADLO 0002
CLR EB44
CNTHI 0001
CNTLO 0000
COMPAR 003E
CRCK EA24
ENHI 003D
ENLO 003C

```

```

GAP 00EB
GP1 00ED
INAX 00EA
INCADR 00D5
LAST 00FB
NSYMH1 000B
NSYML0 000C
NUMA EA46
NXLINE 0078
OUTALL E9BC
PR1A 009B
PRNT1 0096
PRNT2 00A5
PRNT3 00B4
SETADR 00CC
SORT 0024
SRT1 002C
SRT2 002F
START 0010
STHI 003B
STLO 003A
SWAP 0054
SWP1 005C
Y1 0004
Y2 0005
YLIM 0006
YTAB 0007

```

```

CNTLO 0000
CNTHI 0001
ADLO 0002
ADHI 0003
Y1 0004
Y2 0005
YLIM 0006
YTAB 0007
NSYMH1 000B
NSYML0 000C
START 0010
SORT 0024
SRT1 002C
SRT2 002F
STLO 003A
STHI 003B
ENLO 003C
ENHI 003D
COMPAR 003E
SWAP 0054
SWP1 005C
NXLINE 0078
PRNT1 0096
PR1A 009B
PRNT2 00A5
PRNT3 00B4

```

```

SETADR 00CC
INCADR 00D5
INAX 00EA
GAP 00EB
GP1 00ED
LAST 00FB
OUTALL E9BC
CRCK EA24
NUMA EA46
CLR EB44

```

Fig. 4: Source Listing, Block—Move Program

```

; COPY
;
; COPIES ANY-SIZE
; BLOCK OF MEMORY
; TO ANYPLACE IN RAM
;
; BEFORE RUNNING,
; PUT START OF
; BLOCK IN "FROM",
; END OF BLOCK IN
; "END", AND FIRST
; DESTINATION IN
; "TO".
;
; EQUATE LIST
*=$00
FRLO
**++1
FRHI
**++1
ENLO
**++1
ENHI
**++1
TOLO
**++1
TOHI
;
; MAIN PROGRAM
*=$200
START
; INCREMENT "END"
INC ENLO
BNE **+4
INC ENHI
LDY #0
MOVE
LDA (FRLO),Y
STA (TOLO),Y
; INCREMENT "FROM"

```

```

INC FRLO
BNE **+4
INC FRHI
; INCREMENT "TO"
INC TOLO
BNE **+4
INC TOHI
; CHECK IF DONE
SEC
LDA FRLO
SBC ENLO
LDA FRHI
SBC ENHI
BCC MOVE
; ALL DONE
BRK
LAST
BRK
END

```

Fig. 5: Assembly Listing, Block—Move Program

```

==0000
; COPY
;
; COPIES ANY-SIZE
; BLOCK OF MEMORY
; TO ANYPLACE IN RAM
;
; BEFORE RUNNING,
; PUT START OF
; BLOCK IN "FROM",
; END OF BLOCK IN
; "END", AND FIRST
; DESTINATION IN
; "TO".
;
; EQUATE LIST
==0000
      **$00
==0000 FRLO
==0000
      **+1
==0001 FRHI
==0001
      **+1
==0002 ENLO
==0002
      **+1
==0003 ENHI
==0003
      **+1
==0004 TOLO

```

```

==0004
      **+1
==0005 TOHI
;
; MAIN PROGRAM
==0005
      **$200
==0200 START
; INCREMENT "END"
E602 INC ENLO
D002 BNE **+4
E603 INC ENHI
A000 LDY #0
==0208 MOVE
B100 LDA (FRLO),Y
9104 STA (TOLO),Y
; INCREMENT "FROM"
E600 INC FRLO
D002 BNE **+4
E601 INC FRHI
; INCREMENT "TO"
E604 INC TOLO
D002 BNE **+4
E605 INC TOHI
==0218
; CHECK IF DONE
38 SEC
A500 LDA FRLO
E502 SBC ENLO
A501 LDA FRHI
E503 SBC ENHI
90E5 BCC MOVE
; ALL DONE
00 BRK
==0224 LAST
00 BRK
      END
      ERRORS= 0000

```

Fig. 6: Block—Move, Disassembled and Hex Dump

```

<K> **$200
/20
0200 E6 INC 02
0202 D0 BNE 0205
0204 E6 INC 03
0206 A0 LDY #00
0208 B1 LDA (00),Y
020A 91 STA (04),Y
020C E6 INC 00
020E D0 BNE 0212
0210 E6 INC 01

```

```

0212 E6 INC 04
0214 D0 BNE 0218
0216 E6 INC 05
0218 38 SEC
0219 A5 LDA 00
021B E5 SBC 02
021D A5 LDA 01
021F E5 SBC 03
0221 90 BCC 0208
0223 00 BRK
0224 00 BRK

```

```

<M>=0200 E6 02 D0 02
< > 0204 E6 03 A0 00
< > 0208 B1 00 91 04
< > 020C E6 00 D0 02
< > 0210 E6 01 E6 04
< > 0214 D0 02 E6 05
< > 0218 38 A5 00 E5
< > 021C 02 A5 01 E5
< > 0220 03 90 E5 00
< > 0224 00 45 4D 4F

```

KEYBOARD EXPANDOR for APPLE II*

C&H Micro announces the transformation of the APPLE II into a complete upper and lower case system. KEYBOARD EXPANDOR, a hardware-software modification of the APPLE II, actually allows the shift keys to be used just like a conventional typewriter.

The hardware change is a one-wire modification with one solder point. The software is a 1/48 transparent machine language routine which augments the Monitor. All APPLE characters and editing keys are maintained. Cap and Shift Locks and an inverse mode display option included. Compatible with methods displaying ASCII such as Paymar's LCA and Apple's contributed BI 888 CHARACTER GENERATOR. Totally compatible with DOS, allowing use of U/L in TEXT files, PRINT and REN statements, DOS file names, and immediate mode.

KEYBOARD EXPANDOR

\$30.00

TEXTPAGE

\$30.00

Contains KEYBOARD EXPANDOR and an INTERIM BASIC program called TEXTPAGE, which was part of APPLESHIP, the first commercially available product which allowed normal use of the APPLE II shift keys. Recently revised. Gives limited 'page-processing' capabilities: allows entry, editing, storage on disc, and printing (using your own printer driver) of a 'typed page' (55 lines of 80 characters). Requires 24K DOS system.

CHAPPLE

AVAILABLE EARLY 1980

Complete word processor including KEYBOARD EXPANDOR and a full screen editor. We are taking a long time to develop this package because we want the first word processor to use the shift keys to be both powerful and easy to use. KEYBOARD EXPANDOR, TEXTPAGE, and APPLESHIP may be returned for credit toward CHAPPLE.

REAL-TIME CLOCK

AVAILABLE EARLY 1980 for \$5.00

Schematics and software for serially interfacing APPLE II to a real-time clock using approx. \$15.00 in parts and the Gene 1/0.

*APPLE is a registered trademark of Apple Computer Inc.

Full documentation included; software provided on disc (except for Clock). Our products are copyrighted with all rights reserved. Orders, accompanied by certified check or money order, should be sent to:

C&H Micro
P.O. Box 249
CLIFTON PARK, NY 12065

EXCERT, INCORPORATED

• • • AIM - 65 • • •

A SPECIAL NOTE TO OUR CUSTOMERS

- * Thanks to you we have moved to larger quarters.
- * We have also expanded our product offerings and deleted others.
- * Hopefully, we have served your needs and will do so again.
- * We believe a customer is not an interruption of our work, but the purpose of it.

Let Us Serve You Again!

Thanks,

Laurie Root

Vice President

P/N		QTY 1 - 9
A65-1	AIM-65 w/1K RAM	\$375
A65-4	AIM-65 w/4K RAM	\$450
A65-A	Assembler ROM	\$85
A65-B	BASIC ROM	\$100
Special - A65-4AB		
AIM-65 w/4K RAM, Assembler & BASIC ROMs		\$595
Spare Parts (When Available)		
A65-P	Printer	\$40
A65-D	Complete Display Board	\$65
	w/Exchange of Old Board	\$40
A65-K	Keyboard	\$40

P/N		QTY 1 - 9
-----	--	-----------

Power Supplies

PRS3	+ 5V at 3A, + 24V at 1A w/mtg hardware, cord, etc.	\$65
PRS4	+ 5V at 2A, + 24V at .5A w/mtg hardware, cord, etc.	\$50

From The Enclosures Group

ENC1	AIM-65 case w/space for PRS3/PRS4	\$45
ENCL1A	AIM-65 case w/space for PRS3/PRS4 and one expansion board	\$49

Cases with Power Supplies

ENC3	ENC1 w/PRS3 mounted inside	\$115
ENC3A	ENC1A w/PRS3 mounted inside	\$119
ENC4	ENC1 w/PRS4 mounted inside	\$100
ENC4A	ENC1A w/PRS4 mounted inside	\$104

From The Computerist, Inc.

MCP1	Mother Plus™ Dual 44 pin mother card takes MEB1, V1B1, PTC1, fully buffered, 5 expansion slots underneath the AIM	\$80
MEB1	Memory Plus™ 8K Ram, 8K Prom sockets, 6522 I/O chip and programmer for 5V EPROMS with cables	\$200
PTC1	Proto Plus™ Prototype card same size as KIM-1, MEB1, V1B1	\$40
V1B1	Video Plus™ board with 128 char, 128 user char, up to 4K display RAM, light pen and ASCII keyboard interfaces w/cables	\$245

Systems

We specialize in assembled and tested systems made from the above items. Normally, the price will be the total of the items, plus \$ 5.00 for shipping, insurance and handling. Please call or write for exact prices or if questions arise.

Higher quantities quoted upon request.
COD's accepted.
Add \$5.00 for shipping, insurance and handling.
Minnesota residents add 4 % sales tax.

P/N		QTY 1 - 9
-----	--	-----------

From Seawell Marketing, Inc.

MEP2	Little Buffered Mother™ Single 44 pin (KIM-4 style) mother card takes MEB2,PGR2, PTC2 and PIO2. Has on board 5V regulator for AIM-65, 4 expansion slots. Routes A&E signals to duplicates on sides	\$139
	with 4K RAM	\$189
MEB2	SEA 16™ 16K static RAM board takes 2114L with regulators and address switches	
	Blank	\$125
	8K	\$225
	16K	\$325
PGR2	Prommer™ Programmer for 5V EPROMS with ROM firmware, regulators, 4 textool sockets, up to 8 EPROMS simultaneously, can execute after programming	\$245
PIO2	Parallel I/O board with 4-6522's	\$260
PTC2	Proto/Blank™ Prototype card that fits MCP2	\$39
PTC2A	Proto/Pop™ with regulator, decoders, switches	\$99

From Beta Computer

MEB3	32K Dynamic Memory Card w/on board DC to DC converters (5V only .8Amax)	\$419
	with 16K	\$349
	with OK	\$279

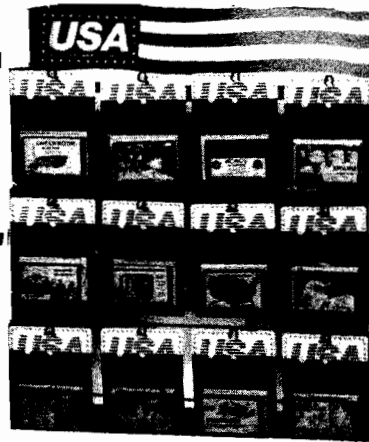
Miscellaneous

TPT2	Approved Thermal Paper Tape 5/165' rolls	\$10
MEM6	6/2114 RAM Chips	\$45

Mail Check or Money Order To:

EXCERT, INC.
P.O. Box 8600
White Bear Lake, MN 55110
(612) 426-4114

GREAT PET SOFTWARE



"Precise, humanized, well documented an excellent value" are the applauds now being given to United Software's line of software. These are sophisticated programs designed to meet the most stringent needs of individuals and business professionals. Every package is fully documented and includes easy to understand operator instructions.

DATABASE MANAGEMENT SYSTEM - A comprehensive, interactive system like those run on mainframes! Six modules comprising 42K of programming allow you to: create, edit, delete, display, print, sort, merge, etc., etc. - databases of up to 10,000 records. Printer routines automatically generate reports and labels on demand. 60 pages of concise documentation are included. Requirements - 16-32K PET and 2040 Dual Disk (printer optional). . . . **Cost \$125**

ACCOUNTS RECEIVABLE/PAYABLE - A complete, yet simple to use accounting system designed with the small businessman in mind. The United Software system generates and tracks purchase orders and invoices all the way through posting "controlled" accounts payable and accounts receivable subsystems.

Keyed Random Access file methods makes data access almost instantaneous. The low-cost solution for the first time computer user with up to 500 active accounts. Requirements - 32K PET, Dual Disk, any 80-column printer. . . . **Cost \$175**

CASH RECEIPTS & DISBURSEMENTS - Makes it a breeze to track all outgoing payments made by any type of business operation. Checks are tracked by number and categorized by type of expense. Sorting, summary, and audit trails make it easy to post to general ledger. This system also categorizes incoming receipts. Uses KRAM file access method. Requirements - 32K PET, Dual Disk (printer optional). . . . **Cost \$99.95**

KRAM - Keyed Random Access Method - The new, ultra-fast access method for the PET Disk, provides keyed retrieval/storage of data, in either direct or sequential mode, by either full or partial key values. Written by United Software in 6502 machine code, and designed with the PET in mind, it exploits all the benefits of the PET Disk, allowing full optimization of your system. Eliminates the need for "Sort" routines! KRAM provides flexibility never seen on a micro before. KRAM is modeled after a very powerful access method used on large-scale IBM Virtual Storage mainframes. So "KRAM" all you can into your PET - it will love you for it. . . . **Cost \$79.95**

(Sublicenses available to software houses.)

PROGRAMS FOR ENTERTAINMENT

Space Intruders	
("Best Game of 1979")	\$19.95
Jury/Hostage	12.50
Kentucky Derby/Roulette	9.95
Alien I.Q./Tank	9.95
Tunnelvision/Maze Chase	14.95
Submarine Attack	9.95
Battle of Midway	7.95
Laser Tank Battle	9.95
Swarm	14.95

Super Startrek	14.95
PET Music Box	29.95

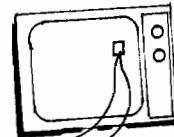
UNITED SOFTWARE PROGRAMS FOR BUSINESS

Checkbook	\$15.95
Mortgage	15.95
Finance	12.95
Bonds	12.95
Stock Analyzer	22.95
Stock Options	24.95
6502 Macro Assembler	49.95

Look for the RED-WHITE-BLUE United Software Display at your local computer dealer, or send check or moneyorder, plus \$1.00 shipping to:

UNITED SOFTWARE OF AMERICA
750 Third Ave.
New York, N.Y. 10017
Dealer inquiries invited

APPLE II SOFTWARE



CURSOR PILOT

gives any Apple II game-paddle control of the video cursor. Activate by touching 'ESC', then edit or copy with game-paddle. Supports normal keyboard controls, is transparent to your programs.

on cassette . . . **\$595**

DATA HANDLER

data base management system. Supports infinite data bases on the Apple II disk drive. Structure data to meet your own needs, up to 255 fields per entry. Advanced data processing allows searching and math to generate reports, extensions, and ledgers. Use for inventory, checks, phone numbers, stocks, lab data, etc. Requires 32K & a disk drive.

on diskette with manual . . . **\$495**

TYPESETTER

a complete HI-RES graphics character generator and editing system. Allows colors, scaling, upper/lower case, inverse, and can HPLOT letters to any point on the screen. Outputs through regular PRINT statements. Use it to label graphs, create ad displays, or print lower case. System includes 35 utility programs and character sets. When ordering, specify if for disk or ROM Applesoft. Needs 32K with ROM, 48K with disk.

on diskette with manual . . . **\$2495**

HIRES UTILITY PACK

Why sweat over HI-RES graphics? Shape Generator lets you build graphic shapes with game paddles, see them at all scales, colors, and rotations. Save them to disk, and Shape Adder puts up to 255 shapes together into a table. Utility Subroutines let you position without plotting, find your last plot, and look at the screen to see if a point is on. Requires 16K with Applesoft ROM.

on diskette . . . **\$1495**

AVAILABLE AT YOUR LOCAL DEALER, OR CALL DIRECTLY AT:

ANDROMEDA COMPUTER SYSTEMS P.O. BOX 19144
GREENSBORO, N.C. 27410
(919) 852-7422



Visa and Mastercard gladly accepted.

Apple II and Applesoft are trade marks of the Apple Computer Company, Inc.

RECYCLE(D) COMPUTERS

BUY ☆ SELL ☆ SWAP

Hardware & Software

NEW PRODUCT ANNOUNCEMENTS

32 pages or more

Mailed 1st Class every 3 Weeks

1yr. (18 issues) ☆ \$3.75

ON_LINE



Dave Beetle, Publisher Established 1975

24695 Santa Cruz Hwy. • Los Gatos, CA 95030

THE BEST WAY TO DETERMINE IF ON_LINE CAN BE OF VALUE TO YOU IS TO TRY A . . .

FREE SAMPLE ISSUE

The MICRO Software Catalogue: XVI

Name: **IRR**
 System: **PET**
 Memory: **16K**
 Language: **BASIC**
 Hardware: **PET(8K) with Cassette**

Description: IRR is designed to provide the potential real estate investor with a detailed breakdown of the projected annual cash flows for the first four years of ownership based on 19 input datum. The second portion of the program provides the projected cash proceeds from the sale, broken down by its various components and tax considerations. The third portion of the program provides the partitioning of the Internal Rate of Return into the three components: Cash Flow, Tax Shelter, and Cash Proceeds from the sale. It then indicates the present value of each component, the percentage of the total return, and the partitioning of the total Internal Rate of Return into the three components. An excellent tool to evaluate prospective real estate purchases.

Copies: **Just Released**
 Price: **\$18.95**
 Includes: **Cassette and Instructions**
 Author: **D.J. Romain**
 Available from:

D. J. Romain, P.E.
 405 Reflection Road
 Apple Valley, MN 55124

Name: **DUNGEON CAMPAIGN**
 System: **APPLE II**
 Memory: **16K (32K for disk version)**
 Language: **Integer BASIC**

Description: Dungeon Campaign is a game of high adventure wherein the player directs an expeditionary force as it ventures into an underground labyrinth. The catacombs are filled with treasures and hazards, poisonous vapors and evil necromancers, stairways and pitfalls, sorcerous devices and in incredible assortment of monstrous inhabitants.

The dungeon's monsters may pursue or wait in ambush. They have a variety of

powers, strengths, and modes of attack, and they become increasingly dangerous in battle as lower levels are reached. As the secrets of the dungeon are uncovered by your force, a color coded map is generated until you find your way safely out with your treasures.

Copies: **Many**
 Price: **\$12.50** cassette, **\$15.00** disk.
 (WA residents add 5.3 percent sales tax)
 Author: **Robert C. Clardy**
 Available: **Synergistic Software**
 5221 - 120th Ave. S.E.
 Bellevue, WA 98006
 (206) 641-1917

Name: **Paper Tiger Graphics Software**
 System: **APPLE II OR APPLE II PLUS**
 Memory: **32K**
 Language: **Integer Basic or Applesoft**
 Hardware: **APPLE II, Disk II, and IDS 440G Printer**

Description: The paper tiger graphics software is a set of programs which allow printing of anything that can be displayed on the Apple II high resolution pages. Any picture, graph, text, or diagram which is displayed can be saved and dumped to the printer. Serial versions of the printing programs are listed on the diskette. The names of the programs indicate which language is used to execute the program. Pictures can be expanded to twice the size and can be inverted to give a black on white or a white on black picture.

Price: **\$34.95**
 Includes: **One diskette plus user pamphlet**
 Author: **David K. Hudson**
 Available: **Local Apple Dealers or Computer Station**
 12 Crossroads Plaza
 Granite City, IL 62040

Name: **Data Handler**
 System: **APPLE II or APPLE II Plus**
 Memory: **32K with ROM — 48K without**
 Language: **APPLESOFT II**

Description: The Data Handler is a data base management system. It can support up to 255 fields/entry. Disk based, it can support infinite data base sizes. Programs allow formatting, editing, sorting, searching, and data processing. Can be used for checkbooks, inventory, stocks, etc. Includes sample files and manual.

Copies: **10**
 Price: **\$49.95** on diskette.
 N.C. residents add 4 percent sales tax.
 Author: **Joe Budge**
 Available: **Andromeda Computer Systems**
 P.O. Box 19144
 Greensboro, N.C. 27410
 (919) 852-1482

Name: **Cursor Pilot**
 System: **APPLE II or APPLE II Plus**
 Memory: **Any Size**
 Language: **Machine**

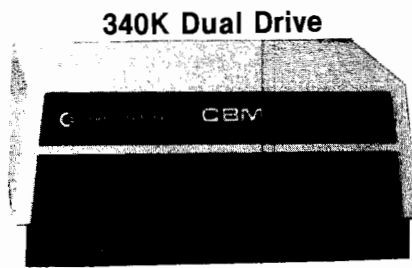
Description: The cursor pilot gives game paddle control of the video cursor. Activate by pressing escape, then edit or copy with the game-paddles. All standard keyboard cursor controls function normally. Transparent to Basic programs. Relocatable program works on any APPLE II with or without disk.

Copies: **Just Released**
 Price: **\$5.95** on cassette
 NC residents add 4 percent sales tax
 Authors: **Joe Budge and Jeff Schmoyer**
 Available: **Andromeda Computer Systems**
 P.O. Box 19144
 Greensboro, NC 27410
 (919) 852-1482

Software Catalog Note

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

Commodore

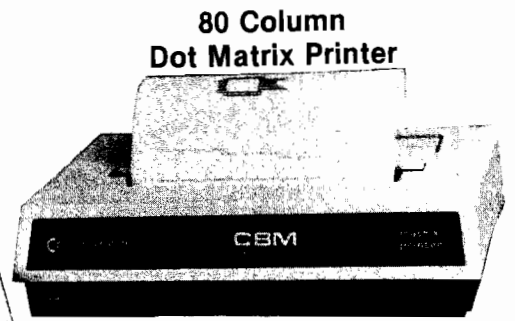


340K Dual Drive

CBM 2040
\$1295⁰⁰



2001 - 32N \$1295⁰⁰



80 Column
Dot Matrix Printer

PRINTERCOM
2022 \$995⁰⁰
2023 \$849⁰⁰

2001 - 8N \$795⁰⁰
2001 - 16B \$995⁰⁰
2001 - 16N \$995⁰⁰
2001 - 32B \$1295⁰⁰
16/32K DIAGNOSTIC KIT \$225⁰⁰
AUDIO AMPLIFIER PET \$29⁹⁵

N DENOTES GRAPHICS ON LARGE KEYBOARD
B DENOTES NO GRAPHICS ON LARGE KEYBOARD

PET to IEEE Cable \$39⁹⁵
IEEE to IEEE Cable \$49⁹⁵
C2N CASSETTE \$95⁰⁰
8K DIAGNOSTIC KIT \$30⁰⁰
DISKETTES:
DYSAN [Business Quality] 5/\$24⁵⁰
VERBATIM 10/31⁹⁵

BUSINESS SOFTWARE

OSBORNE — CMS

General Ledger Disk \$295⁰⁰
Accounts Payable Disk \$195⁰⁰
Accounts Receivable Disk \$195⁰⁰
Word Processor 16/32K Disk \$99⁰⁰

Inventory Control Disk \$195⁰⁰
[Available 12-1-79]
Mailing List Disk \$95⁰⁰
Payroll Disk \$295⁰⁰
[Available 1-15-80]
Word Processor Tape \$24⁹⁵

CBM — MIS

General Ledger Disk \$120⁰⁰
Accounts Receivable Disk \$120⁰⁰
Accounts Payable Disk \$120⁰⁰
Payroll Disk \$120⁰⁰

Inventory Disk \$120⁰⁰
Job Cost/Bid Disk \$120⁰⁰
Customer Information
[Mailing List] Disk \$120⁰⁰

CBM — MIS Complete 7 Module Set \$795⁰⁰

All 16N/16B Upgrade to 32K \$310⁰⁰

Ship computer and check to:

HOME COMPUTERS

1775 E. Tropicana
(Liberace Plaza)
Las Vegas, NV 89109
702/736 - 6363

FREE Software
LAS VEGAS series with any PET
computer purchase or upgrade
to 32K, valued at \$200⁰⁰ or
more, including other software.

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 fashion, 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**

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**

MISSILE-ANTI-MISSILE display a target, missile, anti-missile, a submarine and map of the U.S. on the screen. 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**

TOUCH TYPING TUTOR teaches typing. Indicates speed and errors made. Finger Bldrs, Gen. Typing, Basic Language and User Supplied. Diskette. Written by Wm. A. Massena. **\$19.95**

APPLE MENU COOKBOOK index-accessed data storage/retrieval program. Recipes stored, unlimited lines per entry. Easy editing. Formulated after N.Y. Times Cookbook. Other useful features included. **\$19.95**
Written by Wm. Merlino, M.D.

MAILING LIST PROGRAM maintains complete record of name, address, phone no., mailing labels accommodates parallel card or built-in printer driver, easy data entry. Diskette. 32K. **\$19.95**

POSTAGE AND HANDLING

Please add \$1.25 for the first item and \$.75 for each additional item.

- Programs accepted for publication
- Highest royalty paid

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

SPACE MAZE

STARWARS

ROCKET PILOT Written by Bob Bishop

Each **\$9.95**

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**

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

AC REMOTE CONTROL
FROM YOUR COMPUTER

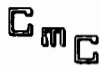
TRS-80 PET S100 APPLE KIM AIM65

INEXPENSIVE CONTROL SOLUTION FOR
HOME SECURITY • ENERGY CONSERVATION
GREENHOUSES • ENVIRONMENTAL CONTROL
INDUSTRIAL CONTROL • LABORATORIES

CmC's μ DAC system now includes an interface to the BSR X-10 remote control modules. These low-cost modules allow control over lamps, motors and appliances. With the CmC X-10 interface your computer can control 256 separate devices. Lamps can be turned on or off, dimmed or brightened. Alarms, kitchen appliances, hi-fis, TVs, motors, pumps, heaters and more can be put under your computer's control.

Direct plug-in and software for most computers.

Circle the reader service number, call or write for our latest catalog.



CONNECTICUT microCOMPUTER, Inc.
150 POCONO ROAD
BROOKFIELD, CONNECTICUT 06804
TEL: (203) 775-9659 TWX: 710-456-0052

STOCK MARKET ANALYSIS PROGRAM DJI WEEKLY AVERAGE 1897-1980

ANA1 (ANALYSIS 1) is a set of BASIC Programs which enables the user to perform analyses on the Dow Jones Industrial weekly average data. From 6 months to 5 years of user selected DJI data can be plotted on the entire screen in one of 5 colors using Apples' High Resolution capabilities. The DJI data can be transformed into different colored graphic representations called transforms. They are: user specified moving averages, a least squares linear fit (best straight line); filters for time, magnitude, or percentage changes; and user created relationships between the DJI data, a transform, or a constant using +, -, x, / operators. Colored lines can be drawn between graphic points. Graphic data values or their dates of occurrence can be displayed in text on the screen. Any graph or text can be outputted to a users printer. The Grid Scale is automatically set to the range of the graphs or can be user changed. As many colored graphs as wanted can be plotted on the screen and cleared at any time. The user can code routines to operate on the DJI/transform data or create his own disk file data base. ANA1 commands can be used with his routines or data base. An Update program allows the user to easily update the DJI file with current DJI weekly data.

The ANA1 two letter user commands are: CA = Calculate, no graph. CG = Clear Graphs, leave Grids. CK = Checking out program, known data. CO = Color of next graph (red, green, violet, white, blue). CS = Clear Screen. DL = Draw Line between points. FI = Filter data for time, magnitude, or percent change. FU = Data, transform, or constant Function with +, -, x, / operator. GD = Graphic mode, display all Graph Data on screen. GR = Graph data to screen. GS = Set Grid Scale. HE = Help, summary of any commands usage. LD = Load Data from disk file from inputted date to memory. LG = Leave Graphs, automatic Grid rescaling. LO = Look, select a range of the LD data and GR; All commands can now be used on this range. LS = Least squares linear fit of the data. MA = Moving Average of the data. NS = No Scale, next graph on screen does not use Grid Scale. NT = No Trace. PR = User implemented Printer routine. TD = Text mode, display Text Data on screen. TI = Time number to date or vice versa. TR = Trace. TS = Text Stop for number of lines outputted to screen when in TD. U1/U2 = User 1/2 implemented routines. VD = Values of Data outputted in text. VG = Values of Grid; low/high/delta. VT = Values of Transform outputted in text.

APPLE® II, 48 K, APPLESOFT
ROM CARD, DISK II DOS 3.2
ANA1 DISK & MANUAL . . . \$49.95
(CA residents add 6% sales tax)

GALAXY
DEPT. M11
P.O. BOX 22072
SAN DIEGO, CA 92122

KIMSI FLOPPY DISKS—

PERRY PERIPHERALS HAS
THE HDE MINIFLOPPY TO KIMSI
ADAPTER

MINIFLOPPY S-100 ADAPTER: \$15

- FODS and TED Diskette
- FODS and TED User Manuals
- Complete Construction Information

OPTIONS:

- FODS Bootstrap in EPROM (1st Qtr'80)
- HDE Assembler (ASM) \$75
- HDE Text Output Processor (TOPS) \$135

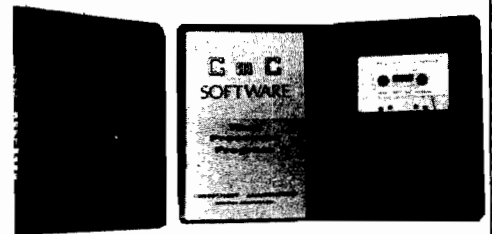
(N.Y. State residents, add 7% Sales Tax)

Place your order with:
PERRY PERIPHERALS
P.O. Box 924
Miller Place, N.Y. 11764
(516) 744-6462

Your "Long Island" HDE Distributor

PET Word Processor

8K
and
16/32K
PET
versions



This program permits composing and printing letters, flyers, advertisements, manuscripts, etc., using the COMMODORE PET and a printer.

Printing directives include line length, line spacing, left margin, centering and skip. Edit commands allow you to insert lines, delete lines, move lines and paragraphs, change strings, save files onto and load files from cassette (can be modified for disk), move up, move down, print and type.

Added features for the 16/32K version include string search for editing, keyboard entry during printing for letter salutations, justification, multiple printing and more.

A thirty page instruction manual is included.

The CmC Word Processor Program for the 8K PET is \$29.50. The 16/32K version is \$39.50.

Order direct or contact your local computer store.

VISA AND M/C ACCEPTED — SEND ACCOUNT NUMBER, EXPIRATION DATE AND SIGN ORDER.
ADD \$1 PER ORDER FOR SHIPPING & HANDLING — FOREIGN ORDERS ADD 10% FOR AIR POSTAGE

CONNECTICUT microCOMPUTER, Inc.
150 POCONO ROAD
BROOKFIELD, CONNECTICUT 06804

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

Search/Change in Applesoft

It is often useful to be able to search a file for a particular string and then to change the string for a new one. This paper presents a Search/Change capability for Applesoft.

J.D. Childress
5108 Springlake Way
Baltimore, MD 21212

A program to produce a cross-reference table for all the variables in a program under development is a useful tool; such a table enables one to determine whether and where a variable label has been used. Unfortunately, a variable's cross-reference program in BASIC is not available in the literature although the development of one was recently reported by William and Alice Englander, *Nybbles: BASIC Cross-Reference Table Generator*, Byte, v4, 4:190 (April 79). About as useful in program development though not as neat for complete documentation purposes is the FIND program of Jim Butterfield, *Inside PET BASIC*, MICRO, 8:39, (December 78-January 79). Butterfield's paper inspired the present SEARCH/FIND program, one that does the same function as Butterfield's but also, allows one to change the found item (within limits).

SEARCH/CHANGE is about seven times as long (1.5 Kbytes) as Butterfield's FIND and runs at about half the speed. It takes about 2.5 minutes to search 8.5 Kbytes. On the plus side, the extra length and sacrifice in speed buys

1. the option not to search or only to search strings,
2. the option to have listed the lines that contain the sought item, and
3. the option to replace the sought item by anything of equal length.

Because of the limitation on length in the CHANGE function, this feature is not really a general purpose program editing tool. Nevertheless, it is quite useful in dressing up variable labels or changing, say, a real variable to an integer variable.

Demonstration

To do a search/change, the SEARCH/CHANGE program must be appended to the program to be searched. Either use the merge feature of the 3.2 DOS renumbering program or the machine language APPEND program and procedure given by Chuck Carpenter, *Renumber Applesoft*, MICRO 12:45 (May 79). Once the programs are wed, enter the search item as line 1 and the change item, if any, as line 2. Then a RUN 63000 starts the works.

To demonstrate the workings of SEARCH/CHANGE, we use the rather nonsensical program listed in Figure 1. We enter the search item DOG as line 1 and run 63000. The print-out of this run is given in Fig. 2. Every appearance of the three consecutive letters D O G is listed. Had we asked for the lines to be listed, a given line would have been listed only once.

We can search for anything; Fig. 3(a) shows the result of a search for equal signs. However, we do have to be careful of Applesoft's reserved words.

Figure 3(b) shows what happens if we try to search for CAT. Applesoft recognizes the reserved word AT in CAT. This makes clear the need of having the program list for verification the search and change items.

The reserved word problem is a relatively minor nuisance. A little ingenuity can get us around it. In the CAT case, we could search for CA; if that gave too many other items, we could then search for TS and only consider the lines that appear in both lists.

The CHANGE function, as well as the line listing feature, is demonstrated in Fig. 4. Again caution is wise. What if we had already used the DGS label in our program? There would be no way later that we could separate the old DGS from the new DGS. If in doubt in changing a label, first make a search to see if the new label is already being used.

In changing the variable label from DOG to DGS, we did not want to change the word DOG inside strings, hence did not search strings. The capability of not searching strings or only searching strings provides all the flexibility we ever need.

We note that we can only change an item to one equal in length (as APPLESOFT sees the length). Extra length in the change item entered as line 2 is ignored. If the replacement is shorter than

the search item, things go awry. The result is a muddle, correctable in general only by a start over from scratch.

Design

A few comments on the design of the SEARCH/CHANGE program are offered here in lieu of remark statements in the program itself.

First the program identifies the search item, FOR loop lines 63040-63070. Then it identifies the change item, if any, FOR loop line 63110 and preceding line. The search is carried out by FOR loop lines 63130-63170. To get the best operating speed, we close the FOR loop within a single line (line 63130) if no byte of significance is found. Even so, the testing for up to three conditions takes time. If one of these conditions is not met, then the following lines either pass to subroutine line 63300 to complete the item identification test and make the item change (if one is entered), or set the string's search flag, or start the search of the next program line, whichever is indicated. Line 63120 determines that the search is over when line 62999 is reached and passes to output. The routine lines 63220-63290 accomplish the line listing feature. Note that the search for the LIST command is backwards from the end of the program (we know that the one we want is the last one). Also note that the line number has to be poked in so that there should always be five digits following LIST. After use of the program, the actual number that appears here when line 63270 is listed is the last number poked in. There should be leading zeros if that number had less than five digits. The Applesoft interpreter preserves these leading zeros whereas the 3.2 DOS renumbering program does not. If you want to renumber SEARCH/CHANGE, remember to check this line and, if you want to, change the 62999 in line 63120.

Figure 1: Listing of Demonstration Program

```

10 FOR I = 1 TO 5
20 PRINT "DOGS AND CATS FIGHT.
   ";: NEXT I: PRINT : PRINT
30 INPUT "GIVE THE NUMBER OF CAT
   S ":CTS: PRINT
40 INPUT "GIVE THE NUMBER OF DOG
   S ":DOG: PRINT
50 IF CTS = 0 AND DOG = 0 THEN END

60 PRINT : PRINT "THE PROBABLE W
   INNER IN A CAT-DOG FIGHT": PRINT
   "WITH ";DOG;" DOGS AND ";CTS
   ";: PRINT "CATS WOULD BE"
70 IF DOG = 0 THEN PRINT "*****
   CATS*****": END
80 IF CTS = 0 THEN PRINT "*****
   DOGS*****": END
90 IF RND (1) * CTS / DOG > .5 THEN
   PRINT "*****CATS*****": END
100 PRINT "*****DOGS*****": END

```

Fig. 2: SEARCH Demonstration

```

)1 DOG
)RUN 63000
1 DOG

PLEASE VERIFY IF THE COMPUTER TAKES
THIS AS YOU INTENDED. DO YOU WANT
TO CONTINUE (YES OR NO)? YES
DO YOU WANT TO SEARCH INSIDE STRINGS
(YES OR NO)? YES
DO YOU WANT TO SEARCH STRINGS ONLY
(YES OR NO)? NO

THE ITEM

1 DOG

IS FOUND IN THE FOLLOWING LINES:

20                40                40
50                60                60
60                70                80
90                100

DO YOU WANT THESE LINES LISTED (YES OR NO)? NO
)

```

Fig. 3: Other SEARCH Demonstrations

(a) Search for equal signs

```

THE ITEM

1 =

IS FOUND IN THE FOLLOWING LINES:

10                50                50
70                80

DO YOU WANT THESE LINES LISTED (YES OR NO)? NO

```

(b) Attempt to search for CAT

```

)1 CAT
)RUN 63000
1 C AT

PLEASE VERIFY IF THE COMPUTER TAKES
THIS AS YOU INTENDED. DO YOU WANT
TO CONTINUE (YES OR NO)? YES
DO YOU WANT TO SEARCH INSIDE STRINGS
(YES OR NO)? YES
DO YOU WANT TO SEARCH STRINGS ONLY
(YES OR NO)? NO

THE ITEM

1 C.AT

IS FOUND IN THE FOLLOWING LINES:

NONE.

```

Fig. 4: CHANGE Demonstration

```
)RUN 63000
1 DOG
2 DGS

PLEASE VERIFY IF THE COMPUTER TAKES
THIS AS YOU INTENDED. DO YOU WANT
TO CONTINUE (YES OR NO)? YES
DO YOU WANT TO SEARCH INSIDE STRINGS
(YES OR NO)? NO
DO YOU WANT TO SEARCH STRINGS ONLY
(YES OR NO)? NO

THE ITEM
1 DOG
IS FOUND IN THE FOLLOWING LINES:
40          50          60
70          90

DO YOU WANT THESE LINES LISTED (YES OR NO)?
YES
THERE WILL BE A WAIT AFTER EACH LINE
UNTIL YOU HIT RETURN TO CONTINUE.

40 INPUT "GIVE THE NUMBER OF DOG
   S ";DGS: PRINT
50 IF CTS = 0 AND DGS = 0 THEN END
60 PRINT : PRINT "THE PROBABLE W
   INNER IN A CAT-DOG FIGHT": PRINT
   "WITH ";DGS;" DOGS AND ";CTS
   ;" CATS WOULD BE"
70 IF DGS = 0 THEN PRINT "*****
   CATS*****": END
90 IF RND (1) * CTS / DGS > .5 THEN
   PRINT "*****CATS*****": END
```

Fig. 5: Listing of SEARCH/CHANGE Program

```
62999 END
63000 DIM SEEK(100),NT(100),L(10
0):START = 256 * PEEK (104)
+ PEEK (103):FINI = 256 *
PEEK (106) + PEEK (105)
63010 IF 256 * PEEK (START + 3)
+ PEEK (START + 2) < > 1 THEN
PRINT "YOU MUST ENTER YOUR
SEARCH ITEM AS LINE": PRINT
"1 BEFORE YOU RUN 63000.": END

63020 LIST 0,2: PRINT "PLEASE VE
RIFY IF THE COMPUTER TAKES":
PRINT "THIS AS YOU INTENDED
. DO YOU WANT": INPUT "TO CO
NTINUE (YES OR NO)? ";Y$: IF
Y$ < > "YES" THEN END
```

```
63030 PRINT "DO YOU WANT TO SEAR
CH INSIDE STRINGS": INPUT "(
YES OR NO)? ";YY$: PRINT "DO
YOU WANT TO SEARCH STRINGS
ONLY": INPUT "(YES OR NO)? "
;YZ$: IF YZ$ = "YES" THEN SQ
= 1:YY$ = "NO"
63040 FOR I = 0 TO 255
63050 SEEK(I) = PEEK (START + 4 +
I)
63060 IF SEEK(I) = 0 THEN N = I -
1: GOTO 63080
63070 NEXT
63080 M = START + N + 6
63090 CH = 0: IF 256 * PEEK (M +
3) + PEEK (M + 2) < > 2 THEN
CH = 1: GOTO 63120
63100 IF N = 0 THEN NT(0) = PEEK
(M + 4): GOTO 63120
63110 FOR I = 0 TO N:NT(I) = PEEK
(M + 4 + I): NEXT
63120 LM = 256 * PEEK (M + 3) +
PEEK (M + 2): IF LM > = 62
999 THEN 63180
63130 FOR I = M + 4 TO M + 255: IF
PEEK (I) < > 0 AND PEEK (
I) < > SEEK(0) AND PEEK (I
) < > 34 THEN NEXT
63140 IF PEEK (I) = 34 AND YY$ =
"NO" THEN SQ = SQ + 1: IF SQ
= 2 THEN SQ = 0
63150 IF PEEK (I) = SEEK(0) AND
SQ < > 1 THEN GOSUB 63300
63160 IF PEEK (I) = 0 THEN M =
I + 1: GOTO 63120
63170 NEXT
63180 HOME : PRINT : PRINT : PRINT
"THE ITEM": PRINT " ";: LIST
1: PRINT "IS FOUND IN THE FO
LLOWING LINES": PRINT : IF
L(1) = 0 THEN PRINT "
NONE.": END
63190 FOR I = 1 TO K: PRINT L(I)
,: NEXT : PRINT
63200 PRINT : INPUT " DO YOU WAN
T THESE LINES LISTED (YES OR
NO)? ";Y$: IF Y$ = "NO" THEN
END
63210 PRINT : PRINT "THERE WILL
BE A WAIT AFTER EACH LINE": PRINT
"UNTIL YOU HIT RETURN TO CON
TINUE.": PRINT
63220 FOR I = 1 TO 1000:W = FINI
- 2 - I: IF PEEK (W) = 188
THEN 63240
63230 NEXT
63240 FOR I = 1 TO K: IF L(I) =
L(I - 1) THEN 63290
63250 L$ = "0000" + STR$ (L(I)):
L$ = RIGHT$ (L$,5)
63260 FOR J = 1 TO 5: POKE W + J
```

```

,48 + VAL ( MID$ (L$,J,1)):
NEXT
63270 LIST 12345: INPUT "";Y$
63280 IF K < 2 THEN END
63290 NEXT : END
63300 IF N = 0 THEN K = K + 1:L(
K) = LM: IF CH = 0 THEN POKE
I,NT(0): RETURN
63310 IF N = 0 THEN RETURN
63320 FOR J = 1 TO N: IF PEEK (
I + J) < > SEEK(J) THEN RETURN

63330 NEXT

63340 K = K + 1:L(K) = LM
63350 IF CH < > 0 THEN RETURN

63360 FOR J = 0 TO N: POKE I + J
,NT(J): NEXT
63370 RETURN

```

Apple-Doc

By Roger Wagner

An Aid to the Development and Documentation of Applesoft Programs

This 3 program set is a must to anyone writing or using programs in Applesoft! It not only provides valuable info. on each of your programs, but allows you to change any element throughout the listing almost as easily as you would change a single line!!

With Apple-Doc you can produce a list of every variable in your program and the lines each is used on, each line called by a GOTO, GOSUB, etc., in fact, every occurrence of almost anything!

You can rename variables, change constants and referenced line #'s, or do local or global replacement editing on your listing.

In fact, we guarantee that after purchase, if you don't feel APPLE-DOC is one of the most valuable programs in your library we will even refund your money! (Upon return of product.)

Unheard of? Yes! But that's how good APPLE-DOC really is!

That's not all! Send for free info. or visit your nearest Apple dealer.

Only \$19.95 Please specify diskette or tape.
(Calif. residents add 6% Sales Tax)

Available from your local computer store or:

Southwestern Data Systems
P.O. Box 582-M
Santee, CA 92071
(714) 562-3670

(Dealer inquiries invited)

BACLAN would like to know if you

WANT TO PROCESS DATA ON YOUR APPLE?

- if so you should be looking for efficient tools to assist with data entry, (i.e. building files) and file handling (i.e. scanning, sorting, printing and copying files).

and

- if you are also looking for economy, we think you will be pleasantly surprised by the low price of the

BACLAN FILE HELPER

available at your Apple Computer Dealer
in both Applesoft and Integer Basic versions



BACLAN P.O. Box 36
(301) 997-9610 Columbia, MD. 21045

MIGHTY BYTE IS HERE

PROGRAM—APPLE II	COMPANY	M F G LIST
Bowling League Secretary	Mighty Byte	24.95
Lisa Interactive Assembler (Req 48K & Disk)	Programma Int.	34.95
Master Catalog (Req 32K & Disk)	Programma Int.	29.95
Apple Pie *Ver 1.0 (Req 32K & Disk)	Programma Int.	24.95
Dr. Memory (Req 32K & Disk)	Muse Co.	49.95
Disk Magic (Req 32K & Disk)	Programma Int.	24.95
Format *Ver 1.0 (Req Disk)	Programma Int.	24.95
3-D Animation	Programma Int.	24.95
Super Dungeon (Req 48K & Disk)	Programma Int.	24.95
U-Draw	Muse Co.	17.95
U-Draw II (Req 32K)	Muse Co.	39.95
Three Mile Island (Req 48K)	Muse Co.	39.95
Escape	Muse Co.	12.95
Tank War	Muse Co.	12.95
Phasor Zap	Programma Int.	15.95
3-D Ducking	Programma Int.	15.95
Strato Laser	Programma Int.	15.95
Depth Charge	Programma Int.	15.95
Super Othello	Programma Int.	15.95
Canter Downs	Programma Int.	15.95
Super Star Wars	Programma Int.	15.95
Many many more programs available.		
Hardware Section		
Centronics 730 Printer		995.00
Heuristics H200 SpeechInk		259.00
Comprint 912 Printer (Parallel)		660.00
Comprint 912 Printer (Serial)		695.00
Heath WH-14 Printer		895.00
Vinyl loose-leaf Diskette Pages (Pack of 10)		8.50

Visa & Mastercharge accepted

To introduce you to MIGHTY BYTE take 15% off any order.

MIGHTY BYTE COMPUTER INC.
P.O.Box 213
HO-HO-KUS, N.J. 07423
(201)445-8256

SYM-1 Staged Loading Technique for Segmented Programs

The SYM cassette tape I/O can not load continuously from 0000 on. The end of page zero and the end of page one can not be directly loaded. A program and technique are presented which simply get around this situation.

Robert A. Peck
P.O. Box 2231
Sunnyvale, CA 94087

The basic SYM-1 comes equipped with 1K of user RAM, most of which can be used for program material. This RAM, however, because of usage by the system monitor, is not contained in a continuous block.

Specifically, the area from roughly 01D1 to 01FF is used as a stack area. Any data or return addresses pushed on to the stack during program (or monitor routine) execution will erase and replace any program material which one might attempt to store in these locations.

Likewise the SYM manual indicates that the page zero locations from 00F0 to 00FF are used occasionally by the monitor program.

Using the SYM tape dump routines, we are able to dump a continuous block 0000 to 03FF to the tape but it is not possible to reload this block in the same manner because of the monitor usage of the areas specified above.

In order to make as full use of the memory space as possible then, we must segment the programs, storing one segment in the area from 0000 to 00EF, another from 0100 to 01CF and the third from 0200 to 03FF (or higher if additional memory is installed).

To store the complete program on tape, we must store the segments independently, since that is the only way we can properly retrieve them. Just as an example, let's say that the first segment has an ID byte of "02", covering 0000-00EF, the second segment an ID byte of "03" (0100-01CF) and third an ID of "04", (extending from 0200 to the end of the program).

Then to reload the program from tape, we must issue three sets of commands, specifically: Load 02 (CR), Load 03 (CR), Load 04 (CR). We must wait for the tape load in between entries. Then we must issue the command which starts the program. If the start location is 0200, we must enter: Go 200 (CR).

It would be much simpler if we were able to enter all of the commands at once and have the machine load all the segments in the right places and then to auto-jump to the start of the program on completion of the load.

Well there is an easy way to set this up with the SYM-1. A 16-byte program entered by the user into any 16 consecutive locations will act as the initial loader program. This is shown in Figure One.

This program would load a program with an ID equal to "01". Because we did a jump to the tape load routine rather than a "JSR", an interesting thing happens. When the tape load routine is done it executes an "RTS", a return from subroutine. This causes the last two bytes pushed onto the stack to be pulled back off and loaded into the program counter.

Therefore when we complete the load of program "01", we will execute a jump to location 0200 because this is the two byte address we pushed onto the stack before the tape load routine was ordered. Program "01" is, in this case, intended to be loaded into locations 0200-0210 and is shown in Figure Two, described below.

This program will load the segment "02" into locations 0-C1, then "03" into locations 100-1C1, and finally segment "04" into locations 0200-03FF. Note that program segment "04" writes over the area where program "01" was loaded. However, since we were under control of the monitor program at the time, it did not matter at all. Besides this, once the third segment is fully loaded, we no longer need the loader program in memory.

After the load, we execute the RTS in the tape loader routine. Since we did not jump to it as a subroutine for the load of the last segment, all it does is to pull 0200 off the stack and uses this as the location of the next instruction to execute.

Therefore by loading those initial 16 bytes in the first program described, we cause the machine to load program 1 which began automatically to load in turn programs 2, 3, and 4. Then it began the execution of our loaded segmented program at location 0200.

The only cautionary note in using this type of sequenced loading is to be certain that the load control segment is located in the area of memory which is overlaid last by the final program segment to be loaded (04 in this case). Otherwise you will erase the loader before the entire group of segments is brought in.

The 16-byte setup program you will note is fully relocatable, and could eventually be linked as a part of your monitor routines. However to make it more general in that case, the instructions

now specified at 020B could be, for example, A5 EE, or reference any other zero page location so that the ID byte could be preloaded there by the user and retrieved by this routine for use later. This also assumes that the user has committed this routine to ROM.

This sequenced loading technique has other uses as well, but that is another subject and may be the subject of a future article.

Figure 1: The Bootstrap Program (Load and start Segment Loader)

0200	20 86 8B	JSR ACCESS	;UNPROTECT SYSTEM RAM
0203	A9 00	LDA #\$00	;STACK LO BYTE OF
0205	48	PHA	;PROGRAM 01 START ADDR.
0206	A9 02	LDA #\$02	;STACK HI BYTE OF
0208	48	PHA	;PROGRAM 01 START ADDR.
0209	A0 00	LDY #\$00	;TAPE MODE (80 IF HI SPD.)
020B	A9 01	LDA #\$01	;PROGRAM ID SEARCHED
020D	4C 78 8C	JMP LOADT	;LOAD PROGRAM 01.

Figure 2: The Segment Loader Program: Loads segments 02, 03, 04 then starts execution at location 0200.

0200	20 86 EB	JSR ACCESS	;UNPROTECT SYSTEM RAM
0203	A9 00	LDA #\$00	;STACK LO BYTE OF PROGRAM
0205	48	PHA	;START ADDRESS
0206	A9 02	LDA #\$02	;STACK HI BYTE OF PROGRAM
0208	48	PHA	;START ADDRESS
0209	A0 00	LDY #\$00	;KIM MODE (80 FOR HI SPD.)
020B	A9 02	LDA #\$02	;PROGRAM ID 02
020D	20 78 EC	JSR LOADT	;JSR TO TAPE LOAD SUBROUTINE
0210	A0 00	LDY #\$00	;TAPE MODE
0212	A9 03	LDA #\$03	;ID 03
0214	20 78 8C	JSR LOADT	;JSR TO TAPE LOAD
0217	A0 00	LDY #\$00	;TAPE MODE
0219	A9 04	LDA #\$04	;ID 04
021B	4C 78 8C	JMP LOADT	;TAPE LOAD *JUMP* , BEGINS
			;PROGRAM AT 0200 WHEN LOAD DONE.

Missing MICRO Information?

MICRO is devoted exclusively to the 6502. In addition, it is aimed at useful, reference type material, not just "fun and games". Each month MICRO publishes application notes, hardware and software tutorials, a continuing bibliography, software catalog, and so forth. Since MICRO contains lots of reference material and many useful programs, most readers want to get the entire collection of MICRO. MICRO grew very rapidly, and it very quickly became impracticable to reprint back issues for new subscribers. In order to make the older material available, two collections of reprints have been published.

A limited number of back issues are still available for number 7 through current.

The BEST of MICRO Volume 1 contains all of the significant material from the first six issues of MICRO, covering October/November 1977 through August/September 1978. This book form is 176 pages long, plus five removeable reference cards. The material is organized by microcomputer and almost every article is included. Only the ads have been omitted.

Surface ... \$7.00
Air Mail ... \$10.00

The BEST of MICRO Volume 2 covers the second six issues, from October/November 1978 through May 1979. Organized by microcomputer, this volume is 224 pages.

Surface ... \$9.00
Air Mail ... \$13.00

Use the convenient Order Form on Page 23 to place your order.

6502 Bibliography: Part XVI

514. Call — Apple 2, No. 5 (June 1979)

- Kotinoff, Jeff, "LORES Color Picture", pg. 19
Two color programs for the Apple II.
- Garson, Dave, "Programmer's Aid Notes", pg. 19
How to use the XDRAW command omitted from the programmer's aid ROM. Two other DEMO programs using the PA ROM of the Apple.
- Golding, Val J., "Book Review", pg. 20
"The Apple II Monitor Peeled" is a very good book by William E. Dougherty, 46 pp \$9.95 available from the author at 14349 San Jose St., Los Angeles, CA 91345
- Aldrich, Darrell, "Scrunch", pg. 21
Discussion and listing of Neil Konzen's program SCRUNCH
- Golding, Val J., "Constructing a Menu", pgs. 25-26
Details of how to put a menu in your program.
- Aldrich, Darrell, "Zero Page Usage by Monitor", pg. 27
A list for Apple Users.
- Lewellen, Tom K., "Integral Data/Parallel Card Fix", pg. 28
Modification of the card solved the problems on the Apple.
- Paymar, Dan, "Prime Factors", pg. 28
A program is listed to compute the prime factors of a given number on the Apple.
- Aldrich, Darrell, "The Apple Doctor", pg. 30
All about the ASCII character set on the Apple.
- Smith, Ken, "HEX/DEC Conversion Program", pg. 30
Convenient Utility Program.
- Ray, R.E., "Fireworks", pg. 31
Two graphics programs.
- Garson, David B., "MOD Function", pg.31
A routine to simulate the "MOD" function in Integer Basic.

515. Contact No. 5 (June, 1979)

- Anon, "Out of the Mist", pgs. 4-6
Subroutine calls for the Apple, Peeks and Pokes
- Anon, "Color Killer Mod for Early Apples", pg. 6
How to modify Apples with serial numbers below 6000.
- Anon, "Shifting Programs from Integer to Applesoft", pg. 6
Routine to automatically shift programs.

516. Interface Age 4, Issue 4 (April 1979)

- Nabers, Steve, "6502 Comprehensive Memory Test Program", pgs. 140-145.
Memory diagnostic set-up for 6502 and implemented on KIM-1.

517. The Computing Teacher 6 No. 4 (May 1979)

- Harder, Monty J., "Bargraph—A Program for the PET Microcomputer," pgs. 45-46.
A simple program for bargraphs — written for ease of adaptation into other programs.

518. The Target, (Jan/Feb 1979)

- Anon, "Binary Indication of the Status Register," pg. 2.
A program for the AIM to print labels for each bit and display the bit in binary.
- Anon, "Bits and Pieces," pg. 3.
Gives info on loading sync characters from tape and lists seven subroutines not included on the AIM Summary Card.
- Anon, "A Program Idea — Soft Memory Expansion," pg. 3
How to get better utilization of your AIM memory.
- Anon, "A Pseudo Waveform," pg. 5
An AIM program to generate a pseudo waveform.
- Anon, "Some of the Printer and Display Routines Explained," pg. 4.
This article supplements the AIM manual in explaining routines.
- Anon, "Disassembly to the User VIA," pg. 5
The program for the AIM gives a quick indication of programs in memory.

519. The Target (May/June 1979)

- Anon, "Symbol Generator," pg. 2.
A symbol generator for the AIM which produces symbols which are user definable.
- Anon, "Enhanced Disassembly to the User VIA," pg. 4.
An extension of the program published earlier. For the Aim.
- Anon, "Sound Generators," pg. 5.
A description of several sound generators for the AIM.
- Riley, Ron, "B.A.P.," pg. 6.
Expand the input/output for the AIM.
- Anon, "Poor or Lazy Man's Regulator," pg. 6.
A simple regulator for the AIM.
- Riley, Ron, "AIM 65 Physical Connections," pg. 7.
Connections for the Display and Printer.

520. Interface Age 4, No. 7 (July 1979)

- Kirschenbaum, Jack, "Need a System Cabinet? Build it!"
Build a cabinet to transport your Apple microcomputer.

521. The Target (Mar/Apr 1979)

- Anon, "AIM 65 Poster," pg. 2.
A program to print a large poster with the AIM.

- Anon, "Software Design—Slow Step," pg. 4
Development of a program for slow stepping the AIM.
- Roland, Don, "AIM 65 Monitor Subroutines," pg. 9.
A numerical listing of the subroutines.
- Riley, R.J., "Regulator Circuits," pg. 9.
Several useful regulators are described.
- Anon, "Using Existing Software," pg. 10-11.
Adapting KIM and other software for the AIM.
- Anon, "Lunar Landing Patch," pg. 11.
Modification of this popular program for the AIM 65.
- 522. Byte 4, No.7 (July 1979)**
- Smith, Stephen B., "Graphic Input of Weather Data",
pg.16-30
Uses an OSI computer and a BIT Pad.
- Bishop, Robert J., "Apple Kaleidoscope," pgs. 52-53.
A fast moving color display for the Apple.
- 523, Creative Computing 5, No. 7 (July 1979)**
- Chatterjee, Rabin, "Picking at 'Peeking and Poking'," pg. 12
Corrections for a previous article (February 1979)
- Petry, Jerry, "Memory Transplants Updated," pg.10
Comments on memory for the PET and TRS-80.
- Friedman, Sl, "Diagnostic Program for Your PET...from
Com-modore," pg. 32-33.
Discussion of the use of several diagnostic routines.
- Kuska, Henry A., "Educational Use of the OSI 1P," pg. 40
Discusses use of a tutor program.
- Milewski, Richard A., "Apple Cart," pg. 116-117.
3-D Graphics on the Apple.
- Yob, Gregory, "Personal Electronic Transactions," pg.
118-122.
Discusses floating point routines and screen gymnastics.
- 524. Abacus Newsletter 1 Issue 6 (June 1979)**
- Anon, "Disc Space Summary," pg. 2
Program for showing sectors available. For the Apple.
- Anon, "Strings and Things," pg. 3.
A routine to concatenate the file name on the end of the
file commands, a routine to find what the first and last
records of the file are, etc. For Apple.
- Anon, "Create Exec Files, It's Easy...", pg. 5.
Program with two examples to help. For Apple.
- Anon, "X-Y Plotter," pg. 6.
Apple program to plot curves.
- Anon, Password Program," pg. 7.
How to secure your Apple programs.
- Anon, "Now You Can Have Lower Case Characters Too," pg.
7
Short program for lower case.
- McCann, Michael J., "How About a (Basic) Disassembler,"
pg. 8-9.
This program will literally take apart a BASIC program and
convert it to machine language. For PET or APPLE II.
- Wilkerson, David, "Lower-Caseing It On The Apple II," pg.
10-11.
Lower case with Integer Basic.
- Bishop, Robert J., "Apple Speaks...Softly," pg. 12-13.
An inexpensive talking Apple II.
- Crossman, Craig, "The Micromodem II," pg. 14.
All about this interesting Modem and the special features
it provides for the Apple.
- Wine, Hal, "Applesoft Stop-List," pg. 15-16.
A short machine language program convenient to use.
- 525. Recreational Computing 8, No. 1, Iss.40 (July/Aug 1979)**
- Fisher, Ted. "Checkmate in Five," pg. 5
Amaze your friends! Beat Peter Jennings's Microchess 1.5
in five moves!
- Lindsay, Len, "How to Fool Around With Your PET," pg.
24-26.
A bouncing ball program with tutorial value.
- Saal, Harry, "SPOT—The Society of PET Owners and
Trainers," pg. 54-55.
New Commodore products for the PET, BASIC Program-
mer's Toolkit, some common basic programs (on tape).
- 526. Apple Peelings 1, No. 1 (July 1979)**
- Anon, "Disk of the Month, July, 1979," pg. 3
The July DOM includes B/BSTAT a version of BINADR
which works with either 3.1 or 3.2 DOS. Apple Peelings is a
new newsletter from the Apple Core of San Francisco and
will alternate on every other month with the CIDER
PRESS.
- 527. Kilobaud Microcomputing No. 32 (August 1979).**
- Lindsay, Len, "PET Pourri," pg. 6-7,12.
New PET ROMs are not compatible with the old ROMs.
Discussion of Skyles new PAL printer, the programmable
character generator, automatic line numberer program,
etc.
- Ascolillo, Carol and Schwartz, Nancy, "Cover Up," pg. 26-37.
Home decoration software for the PET.
- Brown, A.W., "Apple Ciphers," pg. 90-92
The role of the Apple in the development of a medical-
office package.
- Lloyd, Kenny, "Taking AIM," pg. 102-104.
Discussion of the Rockwell International 6502-based AIM
65.
- Hayek Tom, "PET Wrap-up," pg. 110-112.
Haul out the wire-wrap tool and relieve the memory
crunch in your PET.
- Badgett, J. Tom, "Visit to OSI," pg. 118-123.
All you ever wanted to know about OSI.
- 528. MICRO No. 14, July 1979.**
- Carlson, Lt. Robert, USN, "A Baudot Teletype Driver for the
Apple," pg. 5.
Use an expensive Baudot teletype with your Apple.
- Abrahamson, Robert, "Structured BASIC Editor and Pre-
Processor," pg. 7-14.
A versatile preprocessor for the OSI Challenger, makes it
possible to enter, list, modify and resequence BASIC pro-
grams.
- Hertzfeld, Andy, "Intercepting DOS Errors from Integer
BASIC," pg. 17-18.
Integer Basic programs can trap errors from DOS,
diagnose problems, and take remedial action with no in-
tervention from the operator.
- Evans, Melville and Larrowe, Vernon, "AIM Your Spouse
Toward Success at the Supermarket," pg. 19-20.
A grocery list generator. For the AIM.
- Christensen, Alan K., "Boolean Equations Reduced on the
PET," pg. 23-26.
This Basic program trains the PET to perform computer-
aided logic design.
- Mottola, R. M., "Screen Dump to Printer for the APPLE II,"
pg. 27-28.
With this program, print a screen full of information on
your printer after you have reviewed it on the screen.
- Taylor, William L., "OSI Memory Test in Basic," pg. 29.
Find that hidden bug in the many K's of Ram.

SUPER-TEXT™

STANDARD FEATURES

- single key cursor control
- automatic word overflow
- character, word and line insertion
- forward and backward scrolling
- automatic on screen tabbing
- single key for entering "the"
- auto paragraph indentation
- character, word and line deletion
- ditto key
- multiple text windows
- block copy, save and delete
- advanced file handling
- global (multi-file) search and replace
- on screen math and column totals
- column decimal alignment
- chapter relative page numbering
- complete printer tab control
- line centering
- superscripting and subscripting
- two color printing
- underscoring and boldface
- user defined special functions

FAST EDITING

Super-Text was designed by a professional writer for simple, efficient operation. A full floating cursor and multiple text screens facilitate editing one section of text while referencing another. Super-Text's advanced features actually make it easier to operate, allowing you to concentrate on writing rather than remembering complicated key sequences.

FLOATING POINT CALCULATOR

A built in 15 digit calculator performs on-screen calculations, column totals and verifies numeric data in statistical documents.

EXCLUSIVE AUTOLINK

Easily link an unlimited number of on-line files on one disk or from disk to disk. Autolink allows you to search or print all on-line files with a single command. Typical files of items that can be stored in this way include personnel files, prospect files, maintenance records, training records and medical histories.

The Professional Word Processor

for the Apple II

MUSE™

THE LEADER IN QUALITY SOFTWARE

ADVANCED FILE HANDLING

Single key file manipulation and complete block operations allow the user to quickly piece together stored paragraphs and phrases. Text files are listed in a directory with a corresponding index for fast and accurate text retrieval.

PRINTER CONTROLS

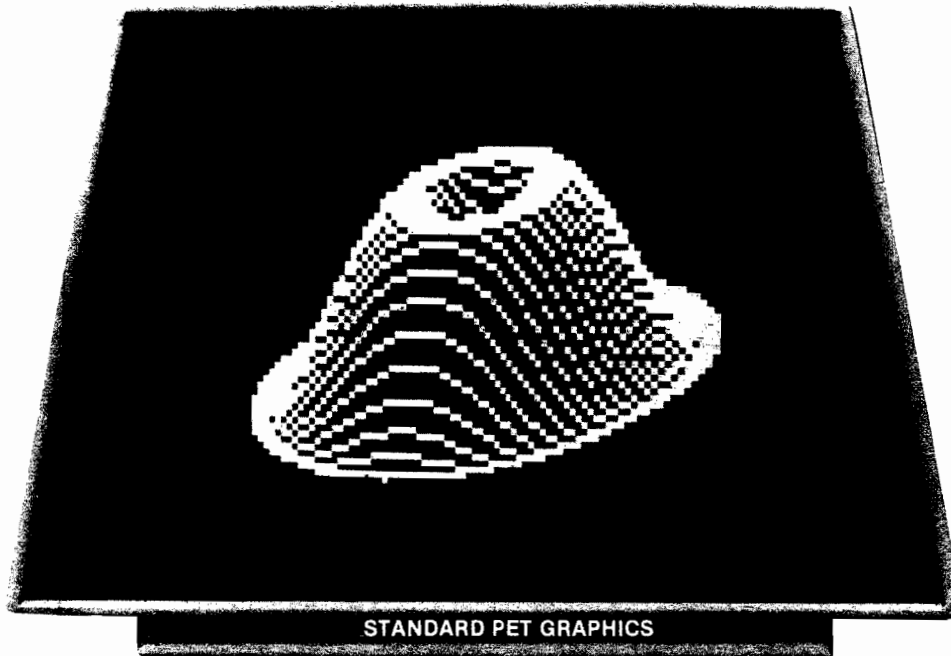
Super-Text is compatible with any printer that interfaces with an Apple. Print single or multiple copies of your text files or link files and they will be automatically printed in the specified order. User defined control characters can activate most special printer functions.

MODULAR DESIGN

This is a modularly designed system with the flexibility for meeting your future word processing needs. The first add-on module will be a form letter generator for matching mailing lists with Super-Text form letters. The form letter module will be available in the first quarter of 1980.

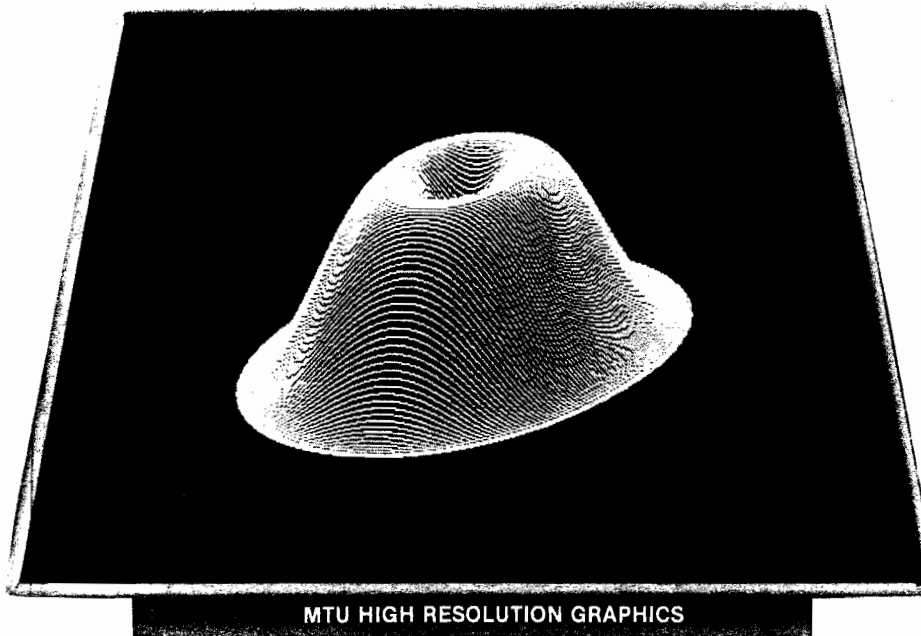
SUPER-TEXT, requires 48K (\$99.95)
Available TODAY at Computer Stores
nationwide. Dealer inquiries welcome. For more
information write:

MUSE SOFTWARE 330 N. Charles Street . Baltimore, MD 21201 (301) 659-7212



HIGH RESOLUTION GRAPHICS

LOOK TO MTU. WE SUPPORT HIGH RESOLUTION GRAPHICS ON:
PET — AIM — KIM — SYM



MTU
Micro Technology Unlimited
P.O. Box 4596, 841 Galaxy Way
Manchester, N.H. 03108
603-627-1464
Call Or Write For Our Full Line Catalog

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, add \$2 shipping. California residents add 6% sales tax. Sorry, we can not ship to P.O. Boxes. VISA/MASTERCHARGE and BANKAMERICARD 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**

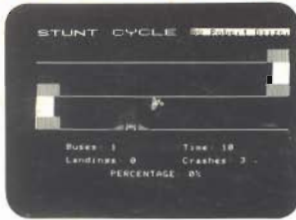


Garden Plaza Shopping Center, Dept. 11A
9719 Reseda Blvd., Northridge, Ca 91324
Telephone: (213) 349-5560

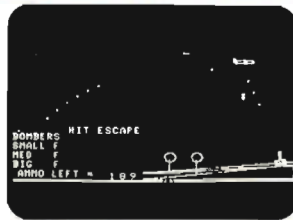


NEW APPLE II SOFTWARE

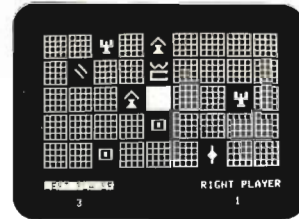
PROGRAMMA Software Program Products



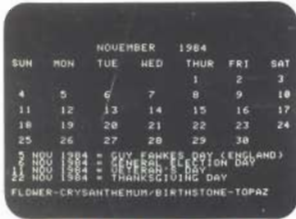
STUNT CYCLE \$15.95



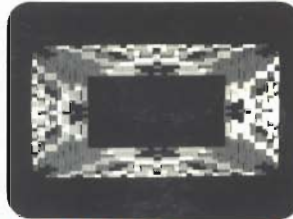
BLITZKRIEG \$15.95



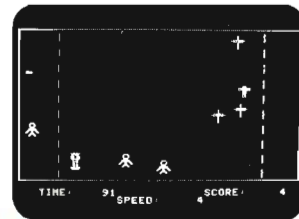
FRUSTRATION \$ 9.95



PERPETUAL CALENDAR \$ 9.95



LORES HYPERPAK \$ 6.95



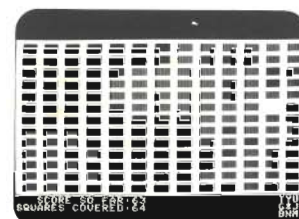
DEATH RACE \$15.95



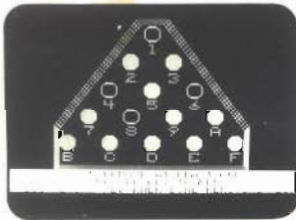
CLOWNS & BALLOONS \$15.95



APPLE ALLEY \$ 6.95



MOUSE HOLE \$ 6.95



PEG JUMP \$ 9.95



I-CHING \$15.95



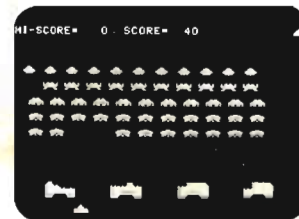
WORDWACKY \$ 6.95



ALIEN INVASION \$ 9.95



GUIDED MISSILE \$15.95



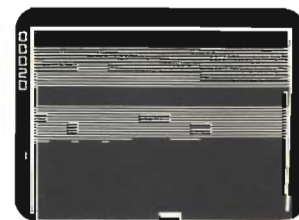
APPLE INVADER \$15.95



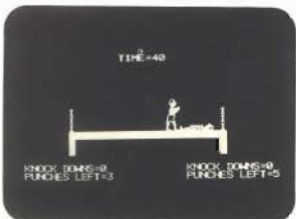
BASEBALL \$15.95

All orders must include 3% postage and handling with a minimum of \$1.00. California residents include 6% sales tax.
 VISA MASTERCHARGE

Apple II is a trademark of Apple Computers, Inc.



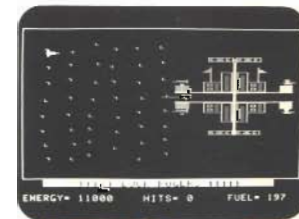
BREAKTHRU \$ 9.95



BOXING \$ 9.95

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

Dealer Inquiries Invited



STAR VOYAGER \$15.95