Computer-Aided Instruction: Boon Or Bust

The 6502 Resource Magazine
PET • Apple • Atari • OSI • KIM • SYM • AIM

Running 40 Column Programs On A CBM 8032

COMPUTE!

$2.50
May, 1981
Issue 12
Vol. 3, No. 5
63379

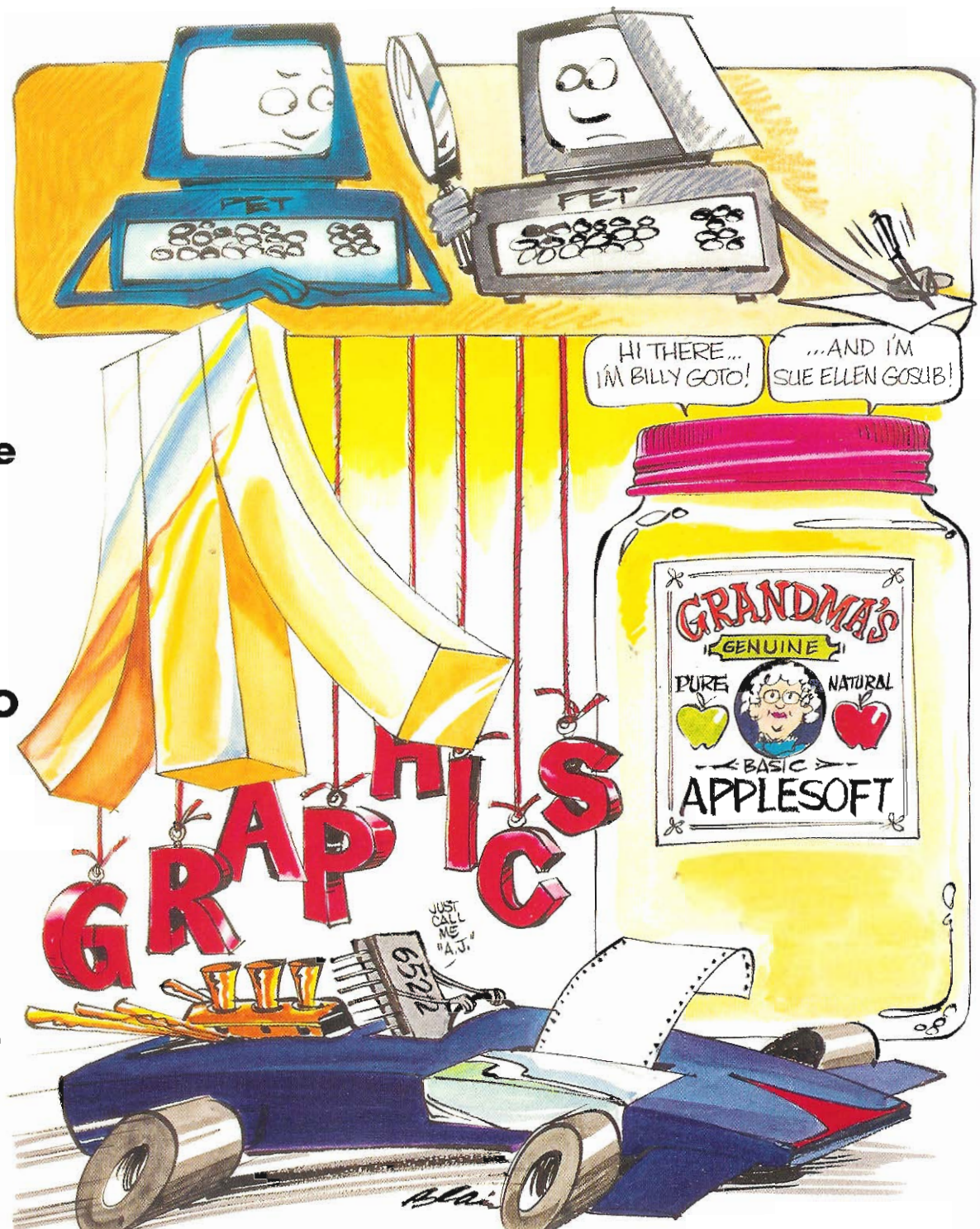
The Journal For Progressive Computing™

PET As An IEEE-488 Logic Analyzer

Using Strings For Graphics Storage On The Atari

Using Named GOSUB And GOTO Statements In Applesoft BASIC

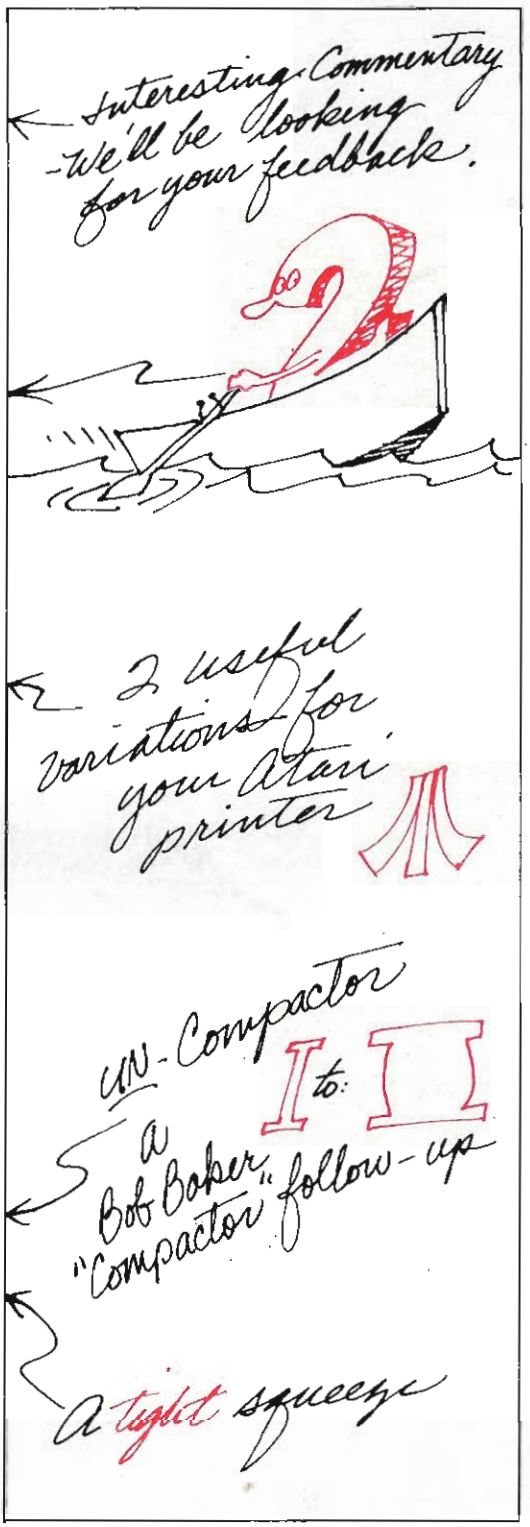
Using The 6522 To Drive A Printer



0 74470 63379 05

Table of Contents

May, 1981, Vol. 3, No. 5



COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published 12 times each year by Small System Services, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 200 East Bessemer Ave., Greensboro, NC 27401.

Domestic Subscriptions: 12 issues, $20.00. Send subscription orders or change of address (P.O. Form 3579) to Circulation Dept., **COMPUTE!** Magazine, P.O. Box 5406, Greensboro, NC 27403. Controlled circulation postage paid at Greensboro, NC 27403. Application to mail at controlled circulation rates pending at Hickory, NC 28601.

ISSN 0194-357X.

A Small System Services, Inc. Publication

COMPUTE! The Journal for Progressive Computing

```
30350 TIME=INT((TIME-SEC)/60)
30400 MIN=TIME-60*(INT(TIME/60))
30500 HOURS=INT((TIME-MIN)/60)
30505 IF SEC>=60 THEN 30510
30508 GOTO 30515
30510 MIN=INT(SEC/60)+MIN
30511 SEC=SEC-60*(INT(SEC/60))
30515 IF MIN>=60 THEN 30520
30518 GOTO 30522
30520 HOURS=INT(MIN/60)+HOURS
30521 MIN=MIN-60*(INT(MIN/60))
30522 IF HOURS=0 THEN HOURS=12
30523 ATCHECK=PEEK(18)*256*256+PEEK(19)*
256+PEEK(20)
30524 IF HOURS>12 THEN HOURS=HOURS-12
30525 SOUND 0,0,0,0
30526 IF ABS(ATCHECK-TOTAT)<100 AND OFF=
0 THEN 30530
30527 GOTO 30539
30530 ? " ":SOUND 0,50,10,10:FOR X=0 TO
1000:NEXT X:? ")))))))"
30539 POSITION 15,10
30540 IF HOURS<10 THEN 30550
30542 IF MIN<10 THEN 30630
30544 IF SEC<10 THEN 30700
30545 PRINT INT(HOURS+0.5);":";INT(MIN+0
.5);":";INT(SEC+0.5):GOTO 30100
30550 IF MIN<10 THEN 30560
30551 GOTO 30600
30560 IF SEC<10 THEN PRINT "0",INT(HOURS
+0.5);":0";INT(MIN+0.5);":0";INT(SEC+0.5
):GOTO 30100
30561 PRINT "0";INT(HOURS+0 5);":0";INT(
MIN+0.5);":";INT(SEC+0.5):GOTO 30100
30600 IF SEC<10 THEN PRINT "0";INT(HOURS
+0.5);":";INT(MIN+0.5),":0";INT(SEC+0:5)
:GOTO 30100
30601 PRINT "0",INT(HOURS+0.5);":";INT(M
IN+0.5),":";INT(SEC+0.5):GOTO 30100
30630 IF SEC<10 THEN PRINT INT(HOURS+0.5
);":0";INT(MIN+0.5),":0";INT(SEC+0.5):GO
TO 30100
30631 PRINT INT(HOURS+0.5);":0",INT(MIN+
0.5);":";INT(SEC+0.5):GOTO 30100
30700 PRINT INT(HOURS+0.5),":",INT(MIN+0
.5);":0";INT(SEC+0.5):GOTO 30100
30900 REM The next lines will poke the h
ardware clock registers down 24 hours
31000 TIME=PEEK(18)*256*256+PEEK(19)*256
+PEEK(20)
31005 TIME=TIME-5184000*(INT(TIME/518400
0))
31020 POKE 18,INT(TIME/(256*256))
31030 TIME=TIME-(256*256)*INT(TIME/(256*
256))
31040 POKE 19,INT(TIME/256)
31050 TIME=TIME-256*(INT(TIME/256))
31060 POKE 20,INT(TIME)
31070 GOTO 30100
```

©

Review Stud Poker

Robert W. Baker
Atco, NJ

STUD POKER is an interesting card game program for the 16K Atari from Dynacomp, Inc., 6 Rippingale Road, Pittsford, NY 14534. ($11.95, cassette; $15.95, diskette) The program includes two separate menu selectable versions of familiar stud poker, each with simple graphics and some sound effects. The card displays are simply the card outline with the face value and suit, no fancy card displays are used. For sound, you get to hear the cards shuffled and dealt along with other appropriate "bells and whistles" at important times.

One of the games deals two cards to you and the Atari, with one card down for the Atari. You each bet on your hands, and bet again after each of the remaining three cards are dealt. At each betting interval you can call, bet/raise from $1 to $3, or fold. The current pot value and your current winnings or loses are always displayed. When the hand is over, the Atari's down card is turned over and the winner is declared.

The other game is even simpler, both you and the Atari are each dealt five cards. Two of the Atari's cards are face down and not displayed. You must bet on your hand ($10 to $100) and cannot fold. After betting, the Atari's down cards are turned over and the winner is declared. Again, your total winnings or loses are displayed.

The games are rather interesting and it would appear that the Atari's card playing skills are pretty good. However, the documentation supplied was rather confusing and did not match the program operation. The names of the two games as well as the betting limits were different in the manual from that used in the program. Also, a different method of indicating whether to continue or quit was used by each part of the program after each hand. One section wanted a "C" or "Q" while the other wanted a RETURN with a null or "Q" input. Totally confusing! With a little more consistency and clearer documentation this could be a very nice package. ©

Un-Compactor

Robert W. Baker
Atco, NJ

Since my Compactor program was published in the Nov./Dec. '80 issue of **COMPUTE!**, I've had several requests for a companion program to un-Compact programs. The program shown here will do just that!

The program reads a BASIC program file from disk on drive 0 and creates a new copy on drive 1. The new program filename is the same as the original except for a "/U" suffix to indicate an un-compacted version. As with Compactor, load the newly created program file and enter a CLR command from the keyboard to correct the program links. Then save the program back to disk as usual. Un-compactor does not generate correct link values when writing the new program file, it merely writes a dummy value to reserve space for a link. This saves a fair amount of extra work not really needed in the program. The CLR command will force BASIC to correct the program links for you.

The program takes any multi-statement lines (statements separated by colons) and breaks them into separate program lines with new line numbers. The new line numbers are generated by adding one to the original line for each new line generated. This procedure is followed for however many statements exist in the line, as long as new line numbers can be generated without reaching the next line number in the original program. If that point is reached, the remainder of the original line is then copied as part of the last line generated with any appropriate separating colons.

The program must take into account certain BASIC tokens or keywords since they effect whether or not a particular line can be broken into separate lines. Thus, any data following a GOTO, END, RUN, IF, RETURN, REM, STOP, LIST, or CONT token is copied unchanged to the end of the current program line. Also, once a quote is detected, the line must be copied until another quote or end of the program line is reached.

Hope this proves to be of help, especially to those currently using Compactor. This program allows you to effectively re-create programs that were compacted. Now you can get a compacted program in Un-compactor to help speed up program execution. As usual, I'll supply copies of the program on cassette for $2 to cover costs.

```
 10 FOR X=1 TO 10
 11 PRINT X
 12 NEXT
 20 PRINT
 21 PRINT
 22 PRINT
 30 REM TEST FILE FOR UNCOMPACTOR
 40 A=1
 41 B=2
 42 C=3
 43 D=4
 44 E=5:F=6:G=7
 45 X=10
 46 Y=20
 47 Z=30
 100 END:THAT ALL!
READY.
```

– SAMPLE LISTING
OUTPUT FILE FROM UNCOMPACTOR

```
 10 FOR X=1 TO 10: PRINT X: NEXT
 20 PRINT:PRINT:PRINT
 30 REM TEST FILE FOR UNCOMPACTOR
 40 A=1:B=2:C=3:D=4:E=5:F=6:G=7
 45 X=10:Y=20:Z=30
 100 END:THAT ALL!
READY.
```

– SAMPLE LISTING
INPUT FILE TO UNCOMPACTOR

```
30 REM          UN - COMPACTOR
50 REM       BY: ROBERT W. BAKER
70 REM 15 WINDSOR DR., ATCO, NJ 08004
100 :
110 GOTO 270
120 :
130 REM >>>>>> SUBROUTINES <<<<<<<
140 :
150 GOSUB 160: V1=V
160 GET#5,C$: GOSUB 190
170 IF C$="" THEN V=0: RETURN
180 V=ASC(C$): RETURN
190 INPUT#15,EN,EM$,ET,ES
200 IF EN=0 THEN RETURN
210 PRINT "ñrDISK ERROR":PRINT
220 PRINT EN;EM$;ET;ES
230 GOTO 1030
240 :
250 REM ***** INITIALIZATION *****
260 :
270 PRINT"ñ";SPC(10);"rUN-COMPACTOR↓↓
280 PRINT" rINPUTf FILE IN rDRIVE #0↓
290 PRINT"rOUTPUTf FILE IN rDRIVE #1↓↓
300 INPUT"rINPUT FILENAMEf";FL$
310 DIM C(256)
320 OPEN 15,8,15
330 OPEN 5,8,5,"0:"+FL$+",P,R"
340 GOSUB 190
350 PRINT:PRINT"OK, WORKING ON LINE# ¬
    ¬.....↓
360 FO$=LEFT$(FL$,14)+"/U"
```

```
370 PRINT#15,"S1:"+FO$
380 OPEN 6,8,6,"1:"+FO$+",P,W"
390 GOSUB 190
400 GOSUB 150: PRINT#6,CHR$(V1);C$;
410 F=1: GOTO 580
420 :
430 REM ***** OUTPUT THIS LINE#
440 :
450 LN=NL: IF LK=0 THEN 1010
460 PRINT LN,
470 PRINT#6,CHR$(1);CHR$(1);
480 PRINT#6,CHR$(LL);CHR$(LH);
490 :
500 REM ***** READ THIS BASIC PGM LINE
510 :
520 X=1
530 GOSUB 160: C(X)=V
540 IF V>0 THEN X=X+1: GOTO 530
550 :
560 REM ***** GET NEXT LINK & LINE#
570 :
580 GOSUB 150: LK=V+V1: IF LK=0 THEN 600
590 GOSUB 150: NL=V1+(256*V): LL=V1:
    ¬ LH=V
600 IF F THEN F=0: GOTO 450
610 :
620 REM ***** BREAK UP LINE IF POSSIBLE
630 :
640 X=1
650 :
660 REM SKIP IF NOT COLON
670 :
680 IF C(X)<>58 THEN 810
690 IF X=1 THEN 950
700 LN=LN+1: IF LN>=NL THEN 950
710 PRINT#6,CHR$(0);CHR$(1);CHR$(1);
720 H=INT(LN/256): L=LN-(256*H)
730 PRINT#6,CHR$(L);CHR$(H);
740 X=X+1: IF C(X)=32 OR C(X)=58 THEN ¬
    ¬740
750 GOTO 680
760 :
770 REM COPY REST OF LINE IF ---
780 REM  GOTO, END, RUN, IF, RETURN
790 REM  REM, STOP, LIST, CONT
800 :
810 IF C(X)<128 OR C(X)>155 THEN 910
820 IF C(X)=128 OR C(X)>153 THEN 850
830 IF C(X)<137 OR C(X)>144 THEN 910
840 IF C(X)=140 OR C(X)=141 THEN 910
850 PRINT#6,CHR$(C(X));
860 IF C(X)>0 THEN X=X+1: GOTO 850
870 GOTO 450
880 :
890 REM SKIP IF NOT QUOTE
900 :
910 IF C(X)<>34 THEN 950
920 PRINT#6,CHR$(C(X)); : X=X+1
930 IF C(X)=34 OR C(X)=0 THEN 950
940 GOTO 920
950 PRINT#6,CHR$(C(X));
960 IF C(X)>0 THEN X=X+1: GOTO 680
970 GOTO 450
980 :
990 REM *** END OF BASIC PROGRAM
1000 :
1010 PRINT#6,CHR$(0);CHR$(0);
1020 PRINT"ĥ⌐DONE":PRINT:PRINT
1030 CLOSE 5: CLOSE 6: CLOSE 15
READY.
```

©