# Formatted Printouts For Commodore

Todd Touris

*Anyone who's written a BASIC program or typed one in from a magazine knows how difficult it can be to decipher the listing. This utility for Commodore computers makes those listings much easier to read. A printer and disk drive are required.*

If you own a printer and a Commodore computer, you probably know how to print out a BASIC program listing. Just load the program into memory, type OPEN 4,4:CMD4: LIST and press RETURN. However, printed listings can be difficult to follow, particularly when program lines contain more than one statement. "Formatted Printouts" is a utility program which improves the readability of BASIC listings, making them easier for you and others to understand.

Type in the program below and save a copy before you run it. It's designed for Epson and Epson-compatible printers. If you have a different printer (Commodore, etc.), minor changes may be needed. The first few lines of the program define several strings for sending control codes to the printer for special modes such as boldface, underlining, and so on. REM statements explain the purpose of each string. Your printer manual should explain which codes to substitute within these strings.

If you're using a VIC-20, you must have at least 8K of memory expansion, and you must also change the following lines:

```
60 POKE36879,8          :rem 10
70 PRINTCHR$(14)"{CLR}{WHT}
   {RVS} PRETTY PRINTER {OFF}"
                        :rem 55
80 PRINT"{8 DOWN}PLEASE WAIT O
   NE MOMENT            :rem 192
130 PRINT"FILENAME TO PRINT":I
   NPUTN$               :rem 146
```

Commodore Plus/4 and 16 users should ignore the :rem statements at the end of each line. These are used with the VIC and 64 "Proofreader" program. Also, with those computers, you need to replace line 60 with this line:

```
60 COLOR 0,1:COLOR 4,1
```

The program is self-prompting and very simple to use. Insert the disk that contains the BASIC program you want to list, then enter the program filename when prompted. That's all it takes. When a program line contains multiple statements, each statement appears on a separate line. Every BASIC keyword (PRINT, GOTO, etc.) is capitalized and printed in boldface. REM lines are underlined, and special graphics characters within quotes are printed as a descriptive string within brackets. For example, the "cursor down" character is printed as **[crsr down]**.

There's one final feature that should be appreciated by those who have used structured languages such as Pascal. All statements inside a FOR-NEXT loop, or after an IF-THEN conditional statement, are indented two spaces, making it much easier to follow the logic of each section. Since this program is written entirely in BASIC, it should not be difficult to add any other features you might desire.

## Formatted Printouts For Commodore

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
10 DIM CHAR$(255),KEYWRD$(75)
                        :rem 77
20 NULL$=CHR$(0):ESC$=CHR$(27)
                        :rem 177
30 S$=ESC$+"E":E$=ESC$+"F":REM
    EMPHASIZED PRINT MODE FOR
   {SPACE}KEYWORDS      :rem 119
40 RS$=ESC$+"-"+CHR$(1):RE$=ES
   C$+"-"+NULL$:REM UNDERLINE
   {SPACE}FOR REM COMMENTS
                        :rem 255
50 SC$="[":EC$="]":REM BRACKET
   S FOR SPECIAL CHARACTER STR
   INGS                 :rem 203
60 POKE53281,11:POKE53280,12
                        :rem 32
70 PRINTCHR$(14)"{CLR}{WHT}
   {RVS}{13 SPACES}PRETTY PRIN
   TER{13 SPACES}{OFF}":rem 55
80 PRINT"{8 DOWN}{8 SPACES}PLE
   ASE WAIT ONE MOMENT..."
                        :rem 108
90 FORL=0TO255:CHAR$(L)=CHR$(L
   ):NEXTL              :rem 1
100 FORL=0TO31:READCHAR$(L):NE
    XTL:FORL=129TO159:READCHAR
    $(L):NEXTL          :rem 180
110 FORL=0TO75:READKEYWRD$(L):
    NEXTL               :rem 243
120 PRINT"{UP}{38 SPACES}"
                        :rem 245
130 INPUT"FILENAME TO PRINT";N
    $                    :rem 6
140 OPEN8,8,8,N$+",P,R":OPEN4,
    4,7:GOSUB290        :rem 134
150 IFLN<>2049THENPRINT"THIS I
    S NOT A BASIC PROGRAM":CLO
    SE8:CLOSE4:GOTO130 :rem 91
160 NSP=0:FOF=0         :rem 119
170 REM MAIN ROUTINE    :rem 199
180 GOSUB290:IFLN=0THEN470
                        :rem 78
190 GOSUB290:LS$=STR$(LN)+" ":
    NSP=NSP-COF:COF=0:LL=LEN(L
    S$):GOTO210         :rem 75
200 LS$="":FORL=1TOLL:LS$=LS$+
    " ":NEXTL           :rem 29
210 GOSUB310            :rem 167
220 IFB>127THENGOSUB380:GOTO26
    0                   :rem 146
```

```
230 IFB=34THENGOSUB330:GOTO270
                          :rem 91
240 IFB$=":"THENA$=A$+B$:GOSUB
    450:GOTO200           :rem 101
250 IFB=0THENGOSUB450:GOTO180
                          :rem 41
260 IFB=167THENGOSUB450:GOTO20
    0                     :rem 145
270 A$=A$+B$:GOTO210      :rem 55
280 REM LINE NUMBER RETRIEVAL
    {SPACE}ROUTINE        :rem 67
290 GET#8,L$:GET#8,H$:LN=ASC(L
    $+NULL$)+ASC(H$+NULL$)*256
    :RETURN               :rem 220
300 REM CHARACTER RETRIEVAL RO
    UTINE                 :rem 216
310 GET#8,B$:B=ASC(B$+NULL$):R
    ETURN                 :rem 76
320 REM QUOTE STRING RETRIEVAL
     ROUTINE              :rem 178
330 IF(B<32)OR((B<160)AND(B>12
    8))THENA$=A$+SC$+CHAR$(B)+
    EC$:GOTO350           :rem 237
340 A$=A$+B$              :rem 47
350 GOSUB310:IF(B=34)OR(B=0)TH
    ENRETURN              :rem 92
360 GOTO330               :rem 104
370 REM KEYWORD INTERPRETER
                          :rem 247
380 A$=A$+S$+KEYWRD$(B-128)+E$
                          :rem 88
390 IFB=167THENCOF=COF+2:GOTO4
    30                    :rem 203
400 IFB=129THENFOF=FOF+2:GOTO4
    30                    :rem 199
410 IFB=130THENFOF=FOF-2:NSP=N
    SP-2:GOTO430          :rem 122
420 IFB=143THENA$=A$+RS$
                          :rem 102
430 B$="":RETURN          :rem 152
440 REM LINE PRINT ROUTINE
                          :rem 87
450 PRINT#4,LS$SPC(NSP)A$+RE$:
    A$="":NSP=FOF+COF:RETURN
                          :rem 95
460 REM END ROUTINE       :rem 123
470 PRINT"FINISHED":CLOSE8:CLO
    SE4:END               :rem 19
480 REM SPECIAL CHARACTER DESC
    RIPTORS               :rem 96
490 DATA"NULL","1","2","3","4"
    ,"WHITE","6","7","SHFTC=OF
    F"                    :rem 126
500 DATA"SHFTC=ON","10","11","
    12","CR","LOWERCASE","15",
    "16","CRSR DOWN"      :rem 228
510 DATA"RVS ON","HOME","DELET
    E","21","22","23","24","25
    ","26","27"           :rem 34
520 DATA"RED","CRSR RIGHT","GR
    EEN","BLUE"           :rem 113
530 DATA"ORANGE","","","","F1"
    ,"F3","F5","F7","F2","F4",
    "F6","F8"             :rem 86
540 DATA"SHFT CR","UPPERCASE",
    "BLACK","","CRSR UP","RVS
    {SPACE}OFF","CLEAR","INSER
    T"                    :rem 199
550 DATA"BROWN","LIGHT RED","G
    RAY 1","GRAY 2","LIGHT GRE
    EN","LIGHT BLUE"      :rem 85
560 DATA"GRAY 3","PURPLE","CRS
    R LEFT","YELLOW","CYAN"
                          :rem 99
570 REM KEYWORDS          :rem 248
580 DATA"END","FOR ","NEXT ","
    DATA ","INPUT#"       :rem 5
590 DATA"INPUT ","DIM ","READ
    {SPACE}","LET ","GOTO "
                          :rem 224
600 DATA"RUN","IF ","RESTORE",
    "GOSUB ","RETURN"     :rem 60
610 DATA"REM","STOP","ON ","WA
    IT","LOAD "           :rem 81
620 DATA"SAVE ","VERIFY","DEF
    {SPACE}","POKE ","PRINT#",
    "PRINT ","CONT","LIST","CL
    R"                    :rem 6
630 DATA"CMD ","SYS ","OPEN ",
    "CLOSE ","GET ","NEW","TAB
    (","TO ","FN ","SPC("
                          :rem 240
640 DATA" THEN"," NOT "," STEP
    "," + "," - "," * "," / "
    ,""                   :rem 149
650 DATA" AND "," OR "," > ","
    = "," < ","SGN "      :rem 60
660 DATA"INT","ABS","USR","FRE
    ","POS","SQR","RND","LOG",
    "EXP","COS","SIN"     :rem 220
670 DATA"TAN","ATN","PEEK","LE
    N","STR$","VAL","ASC","CHR
    $","LEFT$","RIGHT$"
                          :rem 126
680 DATA"MID$","GO"       :rem 128
```
©

# Atari Cassette Verify

Dan Stromberg

*This short, relocatable machine language routine verifies whether a tape save was successful on all Atari 400/800, XL, and XE computers.*

Atari BASIC provides no command for verifying whether a CSAVE or LIST to cassette was successful. "Atari Cassette Verify" remedies that problem. Just type in the program below, save it for future use, and run it. Only a few moments are required to POKE the machine language (ML) routine into memory locations 1644–1746 (near the top of page 6). Since this routine is fully relocatable, you can change the address values in line 100 to whatever other location is convenient. This should be a location which is not erased when other BASIC programs are CLOADed.

Once Cassette Verify is in memory, it's available for use at any time. To verify a program that you've saved, simply type PRINT USR(1664) and press RETURN. (Of course, if you relocate the ML, you should change the 1664 to the new starting address.) You'll hear a buzzing sound, just like the one caused by typing a LOAD command. Position the tape at the file you want to verify, then press any key.

While the ML routine is verifying, you'll hear the usual beeping sounds through the speaker of your TV or monitor. When the operation is complete, the computer prints a number on the screen. If that number is one, the verify was successful—that is, the program on tape matches the program in memory. If you see any other number, consult your BASIC manual to see what the error number means before attempting to resave your program.

## Atari Cassette Verify

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
NF 100 FOR INC=1664 TO 1746
IF 110 READ BYTE:POKE INC,BY
       TE
KM 120 NEXT INC
FA 130 DATA 104,162,48,169,3
       ,157,66,3,169,29,157,
       68,3,169,3,157,69,3,1
       69,1,157,72,3,169
OH 140 DATA 0,157,73,3,169,4
       ,157,74,3,169,128,157
       ,75,3,32,86,228,48,24
       ,169,7,157,66,3
CE 150 DATA 169,0,157,72,3,1
       57,73,3,32,86,228,16,
       238,192,136,208,2,160
       ,1,132,195,132,212,16
       9
HM 160 DATA 0,133,213,169,12
       ,157,66,3,76,86,228
```
©

# ATARI 130XE

**ATARI 130XE Super Computer Package**
130XE Computer
1050 Disk Drive
1027 Printer
Atariwriter
**ONLY $399**

**ATARI PRINTER INTERFACES**
Uprint A ..... 54.95
Uprint AW 16K ... 79.95
Uprint AW 64K ... 99.95
MPP 1150 ..... 59.95

**INDUS GT DISK DRIVE... Call**

**ATARI 130XE SUPER PRINTER PKGS.**
SG-10 Printer and U-Print A ... 279
Panasonic 1091 and U-Print A ... 309
*Super Printer Packages have no extra shipping charges or credit card surcharges when shipped in Continental USA*

## ATARI 130XE SOFTWARE

**BRODERBUND**
Print Shop ..... 28.95
Karateka ..... 20.95
Champ Loderunner .. 23.95
Print Shop Graph. I, II, or III ... 19.95

**INFOCOM**
*See Commodore 64 section for items and prices*

**ELECTRONIC ARTS**
Archon II ..... 24.95
Archon ..... 19.95
Seven Cit. of Gold ... 24.95
Skyfox ..... 24.95
Pinball Const ..... 24.95
One on One ..... 24.95

**MICROPROSE**
Silent Service ..... 23.95
Gunship ..... 23.95
Accrojet ..... 23.95
F-15 Strike Eagle .... 23.95
Kennedy Approach .. 23.95

**OSS**
Basic XE-Cart ..... 52.95
MAC 65 XL-Cart ..... 49.95
Action-Cart ..... 49.95
Basic XL-Cart ..... 39.95
All Tool Kits ..... 20.95

**BATTERIES INCLUDED**
Home Pak ..... 34.95
Paper Clip ..... 34.95
B-Graph ..... 34.95

**SYNAPSE**
Syncalc ..... 32.95
Synfile ..... 32.95
Syntrend ..... 25.95
Syncalc Templates... 16.95
Loderunner Rescue .. 20.95
Mindwheel ..... 27.95
Essex ..... 27.95
Brimstone ..... 27.95

**SSI**
*See Commodore 64 section for items and prices*

**MISCELLANEOUS 130XE**
Hacker ..... 19.95
Amer. Cross Ctry .... 19.95
Flight Simulator II ... 34.95
Ultima II ..... 37.95
Ultima III ..... 37.95
Universe ..... 69.95
Letter Perfect ..... 39.95
Data Perfect ..... 39.95
Halley Project ..... 27.95
Ultima I ..... 23.95
Ultima IV ..... 41.95
MMG Basic Comp ... 69.95

# ATARI 520ST*

**Atari 520ST-RGB System...Call**
**Atari 520ST-Monochrome Sys. . .Call**
**SF314DS/DD 1 Megabyte Disk Drive .....Call**
*We warranty all 520ST computers purchased from ComputAbility for ninety days.*
***Please call for stock availability on Atari ST products before ordering by mail.**

## ATARI 520ST SOFTWARE

**HABA SYSTEMS**
Hippo C ..... 54.95
Haba Write ..... 54.95
Business Letters ..... 34.95
Wills ..... 34.95
Checkminder ..... 54.95

**MISCELLANEOUS ST**
Sundog ..... 27.95
Flip Side ..... 24.95
Softspool ..... 24.95
ViP Professional .... 129.95
Ultima II ..... 39.95
Perry Mason ..... 34.95
Degas ..... 27.95
Farenheit 451 ..... 34.95
Amazon ..... 34.95
Hacker ..... 29.95
The Final Word ..... 94.95
Deja Vu ..... 39.95
Keyboard Cadet ..... 27.95
Halley Project ..... 34.95
PC/Intercom ..... 89.95
Hex ..... 27.95
Chat ..... 19.95
Crimson Crown ..... 27.95
Mudpies ..... 23.95
M-Disk ..... 24.95
Express ..... 34.95

**INFOCOM ST**
Deadline ..... 34.95
Starcross ..... 34.95
Zork I, II or III ..... 29.95
Witness ..... 27.95
Suspended ..... 34.95
Planetfall ..... 27.95
Sorcerer ..... 29.95
Seastalker ..... 27.95
Cutthroats ..... 27.95
Hitchiker ..... 27.95
Suspect ..... 29.95
Wishbringer ..... 27.95
Infidel ..... 29.95
Enchanter ..... 27.95
Spellbreaker ..... 34.95
Mind Forever Voy. ... 34.95

# AMIGA

**AMIGA PERSONAL computer***
**Call for Hardware and add-on peripherals prices**

**AMIGA SOFTWARE**
Archon ..... 27.95
Artic Fox ..... 27.95
Deluxe Paint ..... 59.95
Fin. Cookbook ..... 34.95
One on One ..... 27.95
Seven Cities ..... 27.95
Skyfox ..... 27.95
Hacker ..... 29.95
Mindshadow ..... 29.95
***Please call for availability of Amiga products before ordering by mail.**

EST. 1982

# ComputAbility

We stock hundreds of programs for the Apple, Atari, C-64 and IBM. If you don't see it listed here, don't hesitate to call.

**No Surcharge for MasterCard/Visa.**

VISA MasterCard

## APPLE PRINTER INTERFACES AND BOARDS

Apricorn Parallel w/Graphics .... 69.95
Apricorn 16K Expansion Board ..... 82.95
Apricorn 80 Column Brd. ... 64.95
Apricorn RS232 Interface ..... 69.95
U-Print-Apple IIC w/64K ..... 109.95
U-Print-Apple IIC w/16K ..... 89.95

## LASER 3000 PERSONAL COMPUTER

Apple 2+ Compatible
Built-in Microsoft Basic in ROM
Built-in RGB and Composite Video
Built-in 80 Column Display
Built-in Centronics Printer Interface
Built-in Numeric Keypad
64K RAM
Bundled Productivity Software
Disk Drive
All this and more for the amazing **LOW** price

**$399**

## APPLE SOFTWARE

**BRODERBUND**
Print Shop ..... 33.95
Print Shop Graphics I, II or III ..... 17.95
Print Shop Comp .... 27.95
Karateka ..... 21.95
Carmen Sandiego .... 25.95
Science Tool Kit ..... 39.95
Bank Street Writer .... 44.95
Fantavision ..... Call

**ELECTRONIC ARTS**
Adventure Const .... 34.95
Archon II ..... 27.95
Bard's Tale ..... 29.95
Imagic Football ..... 24.95
Auto-Duel ..... 34.95
Skyfox ..... 27.95
Lords of Conquest ... 27.95
One on One ..... 27.95
Ultima III ..... 39.95
Ultima IV ..... 39.95

**EPYX**
Ballblazer ..... 24.95
Winter Games ..... 24.95
Summer Games II .... 24.95
World's Great/Football ..... 24.95

**INFOCOM**
*See Atari 520ST section for items and prices.*

**MICROPROSE**
*See Atari 130XE section for items and prices.*

**MINDSCAPE**
Color Me ..... 20.95
Crossword Magic .... 34.95
Halley Project ..... 29.95
A View To Kill ..... 27.95
Racter ..... 29.95
The Mist ..... 27.95
Perfect Score ..... 49.95
Voodoo Island ..... 27.95

**DAVIDSON**
Math Blaster ..... 34.95
Alge-Blaster ..... 34.95
Spell It ..... 34.95
Word Attack ..... 34.95

**SIMON & SCHUSTER**
Typing Tutor III ..... 34.95
Kobiashi Adv ..... 29.95
Webster Spell Chk. ... 34.95
Webster Thesaurus ... 84.95
Lovejoy SAT ..... 49.95

**SIR-TECH**
Wizardry/Diam. ..... 23.95
Wizardry/Legacy .... 27.95
Wizardry/Proving ... 33.95
Wizardry/Wernda ... 29.95
Wiziprint ..... 19.95

**SSI**
*See Commodore 64 section for items & prices.*

**APPLE MISCELLANEOUS**
Beach-Head ..... 23.95
Beach-Head II ..... 23.95
Gamemaker ..... 27.95
Hacker ..... 27.95
Hardball ..... 24.95
Sundog ..... 27.95
Paper Clip ..... 64.95
DLM Software ..... Call
The Works ..... 34.95
Star League Base ... 23.95
Educalc ..... 34.95
Friendly Filer ..... 27.95
Note Card Maker .... 27.95
Hayden Software ..... Call
Microleague Base ... 29.95
Random House ..... Call
PFS Software ..... Call
Newsroom ..... 39.95
Gato ..... 27.95
Weekly Reader ..... Call

# C-64 SUPER COMPUTER PACKAGE

C-64 Computer
1541 Disk Drive
803 Printer
**ONLY $399**

# IBM PC

**IBM PC SYSTEMS**
Configured to your specific needs
**Call for lowest price on IBM-PC, IBM-XT or IBM-AT**

**Corona PC-400 Compatible .. Call**
**Corona Portable PC Compatible .. Call**
**PC Multifunction Boards**

We carry the complete line of **AST, Hercules, Paradise, STB, and Quadram**
**Call for current prices**

## IBM PC SOFTWARE

Print Shop ..... 39.95
Print Shop Graph. I ... 27.95
Bank Street Writer ... 49.95
Ancient Art of War ... 29.95

**BORLAND**
Sidekick ..... 37.95
Turbo Pascal ..... 49.95

**BLUE CHIP**
Baron ..... 34.95
Squire ..... 34.95
Millionaire ..... 34.95
Tycoon ..... 34.95

**DIGITAL RESEARCH**
*Call for items and prices.*

**INFOCOM**
*See Atari 520ST for items and prices*

**LEADING EDGE**
Nutshell ..... 69.95
LE/WP Basic ..... 67.95
LE/Word Proc +Speller ..... 169.95

**MICROPROSE**
F-15 Strike Eagle .... 23.95
Kennedy Approach .. 27.95
Acrojet ..... 27.95
Silent Service ..... 27.95

**MICROSOFT**
Flight Simulator ..... 38.95
Word ..... 249.00
Multiplan ..... 134.95

**MINDSCAPE**
*See Apple Section for items and prices.*

**SIERRA**
King's Quest ..... 34.95
King's Quest II ..... 34.95
Ultima II ..... 39.95

**THOUGHTWARE**
*Call for items and prices.*

**IBM MISCELLANEOUS**
PFS ..... Call
Gato ..... 27.95
Wizardry ..... 39.95
Strip Poker ..... 27.95
Electric Desk ..... 204.95
D-Base III ..... Call
Sideways ..... 39.95
Home Pak ..... 34.95
Sargon III ..... 34.95
Peachtree ..... Call
Jet ..... 34.95
BPI Business ..... Call
Newsroom ..... 39.95

## C-64 SUPER PRINTER PKGS.

SG-10 & G-Wiz ..... 279
Panasonic 1091 & G-Wiz ..... 309
Legend 1080 & G-Wiz ..... 282
*Super Printer packages have no added shipping or charge card surcharges added when shipped in Continental USA*

**Westridge AA/AD Modem . . 49.95**
**Cardco G-Wiz . . . 54.95**

## COMMODORE 128

C-128 Computer . Call
1571 Disk Drive .. Call
**Buy both the C-128 & 1571 Disk Drive & receive a 25.00 gift certificate**
1902 Monitor .... Call
1670 Modem .... Call
Westridge 6470 Direct Connect 300/1200 Modem ..... 179.95

## COMMODORE 128 SOFTWARE

Consultant ..... 52.95
Paper Clip ..... 59.95
Swiftcalc w/ Sideways .... 49.95
Wordwriter +Spell ..... 49.95
Data Mgr. II .... 49.95
Fleet System II .. 44.95

## GENERAL HARDWARE

Star Micronics Inc. THE POWER BEHIND THE PRINTED WORD

**SG-10 ..... 215**
**SG-15 ..... 369**
**SD-10 ..... 339**
**SD-15 ..... 449**
**SR-10 ..... 489**
**SR-15 ..... Call**

**PRINTERS**
Panasonic 1091 ..... 245
Legend 808 ..... 169
Legend 1080 ..... 219
Powertype ..... 309
Juki 5510 ..... 389
Epson ..... Call

**PRINTER BUFFERS**
Microfazer ..... From 169
U-Buff 16K ..... 79.95
U-Buff 64K ..... 99.95

**MODEMS**
US Robotics 2400 ..... 469
Volksmodem 1200 ..... 189
Prometheus 1200 ..... 319
Password 1200 ..... 209
Novation ..... Call
PC Modems ..... Call

**MONITORS**
Technika MJ-22RGB .... 269
Sakata SC-100 ..... 169
Commodore 1702 ..... 189
Samsung 12" Green .. 79.95
Samsung 12" Amb. .. 79.95
Taxan ..... Call
Amdek ..... Call

## COMMODORE 64 SOFTWARE

**ACCESS**
Beach-Head ..... 21.95
Beach-Head II ..... 24.95
Raid/Moscow ..... 24.95
Mach V-Cart ..... 21.95

**INFOCOM**
Zork I ..... 24.95
Zork II, or III ..... 27.95
Deadline ..... 29.95
Starcross ..... 29.95
Witness ..... 29.95
Planetfall ..... 24.95
Hitchiker ..... 24.95
Enchanter ..... 24.95
Cutthroats ..... 24.95
Sorcerer ..... 29.95
Spellbreaker ..... 29.95

**SSI**
Battallion Commander ..... 24.95
Battle of Antietnam ..... 32.95
Fighter Command (No Atari) ..... 37.95
Norway 85 (No Atari) ..... 21.95
Panzer Grenidier ..... 24.95
USAAF ..... 37.95
Breakthrough/Ardennes ..... 37.95
Kampgruppe ..... 37.95
Phantasie (No Atari) ..... 24.95
Broadsides ..... 24.95
Carrier Force ..... 37.95
Comp. Ambush ..... 37.95
Mech Brigade (No Atari) ..... 37.95
Field of Fire (No Apple) ..... 24.95
Op. Mkt. Garden ..... 32.95
Pro Tour Golf (No Atari) ..... 24.95
Gemstone Warrior ... 21.95
Imp. Galactum ..... 24.95
Computer Baseball ... 24.95
Comp. Quarterback ... 24.95

**ELECTRONIC ARTS**
Adv. Construction .... 29.95
Mail Order Monster ... 24.95
Racing Destruction ... 24.95
Ultima III ..... 37.95
Ultima IV ..... 41.95
*See Atari 130XE section for rest of items & prices.*

**EPYX**
World/Football ..... 24.95
Eidolon ..... 24.95
Koronis Rift ..... 24.95
Winter Games ..... 24.95
Apshai Trilogy ..... 24.95
Fast Load-Cart ..... 24.95

**MICROPROSE**
*See Atari 130XE section for items and prices.*

**MISCELLANEOUS COMMODORE 64**
Print Shop ..... 28.95
Cal-Kit ..... 34.95
Superbase 64 ..... 47.95
Karateka ..... 20.95
Hacker ..... 20.95
Gamemaker ..... 27.95
Ultima II ..... 37.95
Karate Champ ..... 25.95
Essex ..... 27.95
Kung Fu/Explod Fist ..... 20.95
Kung Fu Master ..... 25.95
9 Princes of Amber ... 22.95
Spy vs. Spy Vol. II ... 23.95
Maxi Golf ..... 24.95
Internat'l Hockey .... 19.95
The Fourth Proto. .... 23.95
Blazing Paddles ..... 24.95
Mirage Word ..... 34.95
Mirage Database .... 34.95
Welcome Aboard .... 19.95
Super Huey ..... 14.95
Spell It ..... 34.95
Math Blaster ..... 34.95
Word Attack ..... 34.95
Odesta Chess ..... 49.95
Brimstone ..... 27.95

EST. 1982

# ComputAbility™

P.O. Box 17882, Milwaukee, WI 53217
ORDER LINES OPEN
Mon-Fri 11 a.m. - 7 p.m. CST • Sat. 12 p.m. - 5 p.m. CST

**To Order Call Toll Free**
**800-558-0003**

**For Technical Info., Order Inquiries, or for Wisc. Orders**
**414-351-2007**

**ORDERING INFORMATION:** Please **specify system.** For fast delivery send cashier's check or money order. Personal and company checks allow 14 business days to clear. School P.O.'s welcome. **C.O.D. charges are $3.00.** In Continental U.S.A. include $3.00 for software orders. 4% shipping for hardware, minimum $4.00. Master Card and Visa orders please include card #, expiration date and signature. WI residents please include 5% sales tax. HI, AK, FPO, APO, Puerto Rico and Canadian orders, please add 5% shipping, **minimum $5.00.** All other **foreign orders add 15% shipping, minimum $10.00.** All orders shipped outside the Continental U.S.A. are shipped first class insured U.S. mail. If foreign shipping charges exceed the minimum amount, you will be charged the additional amount to get your package to you quickly and safely. All goods are new and include factory warranty. Due to our low prices all sales are final. **All defective returns must have a return authorization number.** Please call (414) 351-2007 to obtain an R.A.# or your return will not be accepted. Prices and availability subject to change without notice.

# Apple Keyboard Customizer

Robert Buehler

*With this program you can reconfigure your Apple keyboard and even save the changes on disk for future use. It works on any Apple II-series computer with DOS 3.3.*

Are you frustrated with the Apple keyboard? Are you curious about why Apple arranged the keys in a particular manner? Do you yearn for a numeric keypad? If so, "Keyboard Customizer" may be for you. It lets you rearrange your keyboard any way you want.

For example, you could convert part of the regular keyboard into a numeric keypad—and even make a hexadecimal pad if you desire. This pad can be laid out using the keys of your choice. Do you keep missing the RETURN key and wish it were larger? No problem. Define three keys as RETURNs.

Besides such things as adding a numeric pad, Keyboard Customizer gives you the opportunity to eliminate pet annoyances. For instance, the colon (:) is commonly used when typing Applesoft BASIC programs. As the regular keyboard is set up, the semicolon and colon share the same key. To enter a colon, you must press SHIFT. With Customizer, the positions of these two characters could be reversed.

The question mark is another familiar character for Applesoft programmers as an abbreviation for PRINT. Using Keyboard Customizer, you could reposition the question mark to the semicolon key, making it more accessible. All of these and any other modifications that fit your fancy are at your fingertips with Keyboard Customizer.

## Typing The Program

To prepare Keyboard Customizer, you must type it in with "Apple MLX," COMPUTE!'s machine language entry program found elsewhere in this issue. MLX catches most typing mistakes as they happen and helps assure that you'll finish with an error-free copy. Read the MLX instructions carefully before you begin. When you run MLX, it asks for the starting and ending addresses of the listing you're about to enter. For Keyboard Customizer, respond with these addresses:

**Starting address: 8000**
**Ending address: 81A7**

When you finish typing the listing, MLX prompts you to save a copy on disk.

## Four Customizer Commands

To run Keyboard Customizer, type BRUN KEYBOARD (or whatever filename you specified when you saved the program with MLX). The READY message should appear as usual.

Keyboard Customizer has four commands, which must be preceded by an ampersand (&). Here's a brief summary:

**&0** Restores the keyboard to its original configuration (as does RESET or a reboot).

**&1** Activates the customized keyboard.

**&2** Enters the keyboard editor.

**&3** Prints a list of key values in the format *original key = customized key value*.

All these commands are pretty much self-explanatory except for &2, which calls up the keyboard editor. This is the tool for altering the key values. The first thing you notice after typing &2 is the message FIRST KEY:. This means the program is asking you to begin defining the range of keys you want to customize.

The editor looks at keys sequentially by their ASCII codes. ASCII (American Standard Code for Information Interchange) is a system which assigns numbers to standard characters which appear on teletype and computer keyboards. The ASCII code for an uppercase A, for example, is 65; B is 66; C is 67; and so on. All letters, numbers, punctuation marks, and other symbols have an ASCII code, and a table of these codes can be found in the *Apple II User's Guide* or just about any other computer manual. You can also determine the ASCII value of a character in BASIC by typing PRINT ASC("A"), substituting the appropriate character for A.

To specify a range of keys, first find the ASCII value of the *lowest-numbered* character you want to customize. Enter this value at the FIRST KEY: prompt. Then find the

ASCII value of the *highest-numbered* character you want to customize. Enter this value at the following prompt, which is LAST KEY:. (Therefore, the value you enter at the FIRST KEY: prompt should always be equal to or less than the value you enter at the LAST KEY: prompt.) Any character can begin or end the range, including ESCape or control characters. You'll notice that control characters along with ESCape are displayed in inverse video for easier identification.

After entering the range of keys you wish to edit, you'll see the message ENTER THE NEW REPLACEMENT VALUE FOR EACH KEY. The program displays the first character in the range you specified, followed by a colon. Next, enter the new replacement character. Do not press RETURN—Keyboard Customizer automatically enters a carriage return and then prompts you with the next key to be edited.

When you've assigned new values to all the keys in the range, the program returns to BASIC. Try typing one of the keys you have altered. It should return the reassigned character. Enter a command using that key, or write a program using the key. Even in PRINT and INPUT statements, the key yields its new character value.

## How It Works

It seems as though Keyboard Customizer brings about some drastic changes. Actually, it doesn't. To understand how the program works, let's review how the Apple handles keyboard input.

Every time a key is pressed, Applesoft BASIC looks at memory locations $38–$39, its input hook. These locations normally contain the address of KEYIN ($FD1B), a routine in Read Only Memory (ROM) that gets the keypress from the keyboard. However, the input hook can be made to point to an alternate input routine. This is the case with Keyboard Customizer. Control passes not to the KEYIN routine in ROM, but rather to a routine within Customizer. This routine calls KEYIN to get the character code for the keypress, but checks to see if the code belongs to a character that was altered. If so, Customizer replaces it with the customized value.

The part of Customizer which replaces the old key values is actually very short (only five bytes). A much larger part of the program is the buffer it uses to store the modified values. Along with the editor, the buffer comprises the majority of the program. The buffer is so large because it stores the values for all the keys sequentially, even if they equal the original values. As a result, the buffer size is constant: half a page of memory (128 bytes). It may seem like a waste of memory to store the values of keys which haven't been changed. But if only modified keys were stored in the buffer, the routine that replaces the character values would be much longer and more complicated.

This brings up another important point. Keyboard Customizer's improvements are temporary, since the input hook at $38–$39 is initialized during a reset or reboot. But there's a way to save the keyboard changes you've made. First, enter the Apple's built-in machine language monitor by typing CALL−151. Then type this line and press RETURN:

```
8016: EA EA EA
```

This stops Keyboard Customizer from clearing the buffer by overwriting three machine language instructions with NOPs (No Operation, similar to REM in BASIC). Second, you'll need to save the buffer that holds all the modifications, along with the original program. Enter this command:

```
BSAVE KEYBOARD1,A$8000,L$23C
```

To run this new version, simply type BRUN KEYBOARD1. You could also include the command BRUN KEYBOARD1 in the HELLO program so the customized keyboard automatically loads every time you boot the system.

### Keyboard Customizer Routines and Important Locations

| | | |
|---|---|---|
| **AMPERV** | $3F5 | Holds JMP instruction to S/R for & commands |
| **CH** | $24 | Cursor horizontal displacement |
| **COUT** | $FDF0 | Prints byte in accumulator on screen |
| **CRDO** | $DAFB | Prints a carriage return |
| **CV** | $25 | Cursor vertical position |
| **DOSHOOK** | $3EA | Connects I/O hooks to DOS |
| **GETBYT** | $E6F8 | Evaluates formula at TXTPTR |
| **KEYIN** | $FD1B | Gets next key input from keyboard |
| **KSWL** | $38–$39 | DOS input hook |
| **RDKEY** | $FD0C | Call KEYIN via KSWL |

Please refer to the "MLX" article in this issue before entering the following listing.

### Apple Keyboard Customizer

```
START ADDRESS:  8000
END ADDRESS:    81A1

8000: A2 4C A0 2F A9 80 8E F5 CE
8008: 03 8C F6 03 8D F7 03 A0 AF
8010: 4C A9 81 20 3B 81 20 2B 1F
8018: 81 20 6F FB A0 27 A9 80 05
8020: 84 38 85 39 4C EA 03 20 E9
8028: 1B FD A8 B9 21 81 60 20 D6
8030: F8 E6 E0 00 D0 0A A0 57 CA
8038: A9 81 20 3B 81 4C 19 81 17
8040: E0 02 D0 69 20 19 81 A0 EB
8048: 86 A9 81 20 3B 81 20 0C 55
8050: FD 8D 9E 81 20 23 81 20 50
8058: F0 FD A0 91 A9 81 20 3B 4D
8060: 81 20 0C FD 8D A0 81 20 9D
8068: 23 81 20 F0 FD A0 5C A9 43
8070: 81 20 3B 81 20 19 81 AD CF
8078: 9E 81 20 23 81 20 F0 FD CB
8080: A9 BA 20 F0 FD A9 A0 20 10
8088: F0 FD 20 0C FD 8D 9F 81 2D
8090: 20 23 81 20 F0 FD AC 9E 14
8098: 81 AD 9F 81 99 21 81 20 46
80A0: FB DA CC A0 81 F0 60 EE 79
80A8: 9E 81 4C 77 80 E0 03 D0 B8
80B0: 4B 20 58 FC 20 FB DA AD 8E
80B8: 9F 81 85 24 AD 9E 81 20 E7
80C0: 23 81 20 F0 FD A9 BD 20 F8
80C8: F0 FD AE 9E 81 BD 21 81 48
80D0: 20 23 81 20 F0 FD EE 9E D8
80D8: 81 20 FB DA AD 9E 81 C9 84
80E0: DF F0 24 A5 25 C9 14 D0 36
80E8: CE A9 00 85 25 A9 08 18 0C
80F0: 6D 9F 81 8D 9F 81 20 FB D8
80F8: DA 4C B7 80 A0 54 A9 81 A4
8100: 20 3B 81 4C 1C 80 60 A9 24
8108: 00 8D 9F 81 A9 80 8D 9E 83
8110: 81 A9 DE 8D A0 81 4C 1C B2
8118: 80 20 89 FE 20 93 FE 20 F1
8120: EA 03 60 C9 A0 10 03 38 85
8128: E9 80 60 A9 80 A2 00 9D 13
8130: A1 81 E8 A8 C8 98 C9 FF 48
8138: D0 F5 60 84 06 85 07 A0 6A
8140: 00 B1 06 F0 06 20 F0 FD 10
8148: C8 D0 F6 60 8D D2 C5 C1 CD
8150: C4 D9 8D 00 CF CE 00 CF 67
8158: C6 C6 87 00 8D 8D C5 CE 5E
8160: D4 C5 D2 A0 D4 C8 C5 A0 99
8168: CE C5 D7 A0 D2 C5 D0 CC 65
8170: C1 C3 C5 CD C5 CE D4 8D 7B
8178: C6 CF D2 A0 C5 C1 C3 C8 BC
8180: A0 CB C5 D9 8D 00 C6 C9 E0
8188: D2 D3 D4 A0 CB C5 D9 BA 72
8190: 00 A0 A0 A0 CC C1 D3 D4 C3
8198: A0 CB C5 D9 BA 00 80 00 0C
81A0: DE 00 00 00 FF FF 00 00 13
```

©

# IBM Advanced Function Key Techniques

Peter F. Nicholson, Jr.

*Restoring original key definitions, extending definitions for certain keys beyond the default limits, and saving definitions to disk for later use are among the techniques covered in this revealing article. For the IBM PC and PCjr and most compatibles.*

Anyone who has ever redefined the function keys in an IBM BASIC program probably has wondered why there's no command to restore the keys' original definitions when the program ends. Usually you end up disabling them or redefining them again to their default values. But there is an alternative, and the secret lies within something called the *soft key buffer*. Locating and examining this buffer can yield some interesting results.

Finding the buffer is easy if you have an IBM PC, XT, or PCjr. It starts at memory location 1619 in the default memory segment. But this is not necessarily true if you have an IBM-compatible computer. Therefore, if you're using a compatible, you should run Program 1. This program attempts to locate the soft key buffer for you. When you find it, you should alter the buffer address (1619) in the IBM programs before running them on your compatible. The lines where this address can be found are indicated in REMark statements within each program.

## Saving Key Definitions

The soft key buffer is just a section of memory which stores the definitions for the function keys. When a key is assigned a different function, its definition within the buffer is altered. A key definition can contain up to 15 characters. If you PEEK into the buffer's memory locations, you may be surprised to find that each key is assigned not 15, but 16 positions. We'll explain why in a moment. In the meantime, knowing the number of positions allotted for each function key makes it easy to save the buffer's contents—and therefore to preserve the keys' definitions.

Program 2 does this by reading the contents of the buffer into an array. Then it assigns new functions to the keys (nonsense definitions for this example). Finally, the program lets you restore the original functions by POKEing the contents of the array back into the soft key buffer. You can use this technique in your own programs to restore the function keys.

Now, if you're still wondering why each key is assigned 16 positions in the buffer when its definition can be only 15 characters long, disabling the keys will provide the answer. If you PEEK at the 16 positions reserved for F1 (originally defined as LIST) and print out the ASCII values, this is what you'll see:

```
L I S T 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

When you disable F1, the buffer looks like this:

```
0 I S T 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

This seems to indicate that BASIC marks the end of a function key definition with a zero. To prove this, run Program 3. It demonstrates that you can restore the function keys after disabling them by merely saving the first character of each key definition (assuming, of course, that the keys have been disabled by overwriting only the first character of the definition). That's why Program 3 needs to save only 10 bytes instead of the 160 bytes saved by Program 2.

## Extended Definitions

Knowing that you can restore the disabled function keys by saving only the first character of each definition may be interesting, but the difference between 10 and 160 bytes probably is of little concern to you. The real power in this knowledge is that you can extend the number of characters available for a function key's definition by altering the sixteenth position in the buffer for that key. This lets you assign a longer definition to a function key (at the expense of the following key, however).

For instance, I prefer to edit programs in SCREEN 0,0,0 and WIDTH 80. Using Program 4, I can set F9 to execute these commands even though they exceed 15 characters. F10 becomes useless, since we haven't increased the size of the soft key buffer—just the length of F9's definition within that buffer.

Program 4 also lets you save the new function key definitions as

# Finally, something Apple and IBM owners can agree on:


# The Sider™ 10 MB hard disk

Only $595 Thru December 31, 1985

***from First Class Peripherals***

Decisions, decisions. First you had to choose between Apple and IBM. Now you have to decide which hard disk subsystem to purchase—and they all seem about the same. *But are they?*

First Class Peripherals can make your hard disk decision a lot easier. Because whether you use an Apple II+ or IIe...or IBM PC* or XT...we offer a Sider 10 MB hard disk subsystem just right for all your storage needs.

### The most reliable, affordable 10 MB hard disk on the market

The Sider features state-of-the-art Winchester disk technology. Direct booting without floppies. Self-contained power supply. And compatibility with the most popular Apple or IBM software.

In addition, the Sider is *plug and play.* Everything you need for quick, easy installation is included: cable, host adapter, software and manual.

### Built to last by Xebec

The Sider has won rave reviews for its performance and reliability. That's because it's manufactured exclusively for First Class Peripherals by Xebec, the industry's leading manufacturer of computer disk drives and controllers. And it's sold *direct to you,* so there are no dealers or distributors to hike up the cost.

### Full guarantee and free tech hotline

You can choose your Apple or IBM Sider with confidence. Simply order and use your Sider for 15 days. If you're not 100% satisfied, return it for a full refund. Keep it, and you'll enjoy a full one-year limited warranty...plus access to our toll-free hotline, should you ever have a technical or service question.

### It's easy to order your Sider

The Sider is priced at just $595 for the Apple model...$595 for the IBM. *That's hundreds of dollars less than what you'd expect to pay for the comparable "big name" models.* To order, use the coupon below...or for faster service, order by phone using Visa, MasterCard or American Express. Call toll-free:

## 1 800 538-1307

**Extension 702**

*Must contain hard disk ROM.

FIRST PERIPHERALS CLASS

**3579 Highway 50 East, Carson City, NV 89701**

☐ Yes, please send me the Sider, including 10 megabyte hard disk drive, host adapter card, cable, complete installation software and documentation for my: ☐ **Apple II+ or IIe** ☐ **IBM PC or XT**

I prefer to pay as follows:

☐ I've enclosed my check or money order for $595* plus $15 shipping and handling, payable to First Class Peripherals.

☐ Please bill the following credit card account for $595* plus $15 shipping and handling:

☐ VISA ☐ MasterCard ☐ American Express

*Card #* *Exp. Date*

*Signature*

*Residents of CA, NV and PA, please add appropriate sales tax.

*Name*

*Address*

*City*

*State* *Zip*

*Telephone (area code)*

**Mail to: First Class Peripherals**
**3579 Highway 50 East, Carson City, NV 89701**

702

a file which can be BLOADed from another program. If you try this, don't omit the buffer address (1619) when BLOADing the file, since there is no way to insure that BASIC's segment will be the same as when you originally created the file.

For instructions on entering these listings, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

## Program 1: Buffer Finder For Compatibles

```
PC 100 DEF SEG:SCREEN 0:WIDTH 80
       :X=0
OH 110 CLS:PRINT "MEMORY LOCATIO
       N ";:LOCATE ,20
IO 120 KEY 1,"LIST":A=ASC("L")
DG 130 IF PEEK(X)=A THEN GOSUB 1
       50 ELSE PRINT X;:LOCATE 1
       ,20
JG 140 X=X+1:GOTO 130
OM 150 IF CHR$(PEEK(X+1))<>"I" T
       HEN RETURN
PG 160 IF CHR$(PEEK(X+2))<>"S" T
       HEN RETURN
CG 170 IF CHR$(PEEK(X+3))<>"T" T
       HEN RETURN
JK 180 CLS:PRINT "MEMORY LOCATIO
       N ";X
NH 190 FOR J=1 TO 10:PRINT "F";J
       ;:FOR K=0 TO 15
AH 200 IF PEEK(X+16*(J-1)+K)>0 T
       HEN PRINT CHR$(PEEK(X+16*
       (J-1)+K)); ELSE 220
OO 210 NEXT K
PP 220 PRINT:NEXT J
GG 230 BEEP:INPUT "IS THIS IT ";
       Q$
ND 240 IF Q$="Y" OR Q$="y" THEN
       END ELSE X=X+1:CLS:GOTO 1
       10
```

## Program 2: Restoring Function Definitions

```
HO 90 REM LINES WHICH USE 1619 O
      FFSET ARE 140 AND 250
DB 100 SCREEN 0:WIDTH 80:CLS:DEF
       SEG:OPTION BASE 1
QM 110 KEY ON:DIM K$(10):FOR X=1
       TO 10:K$(X)=STRING$(16,0
       ):NEXT X:' STORAGE AREA F
       OR FUNCTION KEYS
BG 120 REM SAVE FUNCTION KEYS
HL 130 FOR X=1 TO 10:FOR J=0 TO
       15
EH 140 MID$(K$(X),J+1,1)=CHR$(PE
       EK(1619+16*(X-1)+J))
OP 150 NEXT J,X
HA 160 REM REDEFINE FUNCTION KEY
       S WITH LETTERS (THIS IS O
       NLY AN EXAMPLE)
JC 170 FOR X=1 TO 10:KEY X,CHR$(
       X+64):NEXT X:KEY LIST
EO 180 PRINT "Function keys are
       redefined":PRINT "Press a
       ny key to restore"
NP 190 KB$=INKEY$:IF KB$="" THEN
       190
PF 200 REM RESTORE FUNCTION KEYS
OF 210 FOR X=1 TO 10
CC 220 KEY X,K$(X)
HM 230 NEXT X:CLS
PL 240 FOR X=1 TO 10
ME 250 J=ASC(MID$(K$(X),16,1)):I
       F J>0 THEN POKE 1619+16*(
       X-1)+15,J
HC 260 NEXT X:CLS
EL 270 KEY LIST
```

## Program 3: Restoring Function Definitions

```
OL 90 REM LINES WHICH USE 1619 O
      FFSET ARE 140 AND 220
EK 100 SCREEN 0:WIDTH 80:CLS:DEF
       SEG
OG 110 KEY ON:K$=STRING$(10,0):'
       STORAGE AREA FOR FUNCTIO
       N KEYS
BG 120 REM SAVE FUNCTION KEYS
PI 130 FOR X=1 TO 10
DJ 140 MID$(K$,X,1)=CHR$(PEEK(16
       19+16*(X-1)))
GH 150 NEXT X
NJ 160 REM DISABLE FUNCTION KEYS
HE 170 FOR X=1 TO 10:KEY X,"":NE
       XT X:KEY LIST
MA 180 PRINT "Function keys are
       disabled":PRINT "Press an
       y key to restore"
NP 190 KB$=INKEY$:IF KB$="" THEN
       190
PF 200 REM RESTORE FUNCTION KEYS
OF 210 FOR X=1 TO 10
PO 220 POKE 1619+16*(X-1),ASC(MI
       D$(K$,X,1))
HM 230 NEXT X:CLS
DF 240 KEY LIST
```

## Program 4: Extending Definitions

```
OC 90 REM LINES WHICH USE THE 16
      19 OFFSET ARE 180,290,390,
      440,470
IF 100 DEF SEG:STK$=STRING$(128,
       0):SCR$=STRING$(37,0):RES
       TORE 110:FOR X=1 TO 37:RE
       AD J:MID$(SCR$,X,1)=CHR$(
       J):NEXT X:SCR!=PEEK(VARPT
       R(SCR$)+1)+256*PEEK(VARPT
       R(SCR$)+2)
LG 110 DATA 85,137,229,139,118,6
       ,41,192,138,4,139,116,1
NB 120 DATA 1,240,137,196,184,0,
       6,187,0,7,185,0,2
FP 130 DATA 186,80,24,85,205,16,
       92,93,202,2,0
CE 140 SCREEN 0:WIDTH 80:CLS
HI 150 T$="Function Key Definiti
       on"
FH 160 LOCATE 2,(40-.5*LEN(T$)):
       PRINT T$
QP 170 PRINT:PRINT
HE 180 X=1:J=1:K=1619
EF 190 K$=STRING$(160,0):KN$=STR
       ING$(160,0):K=K-1
CP 200 L=PEEK(J+K)
PN 210 WHILE L<>0
PH 220 MID$(K$,J,1)=CHR$(L)
OK 230 J=J+1:L=PEEK(J+K)
EN 240 WEND
ID 250 PRINT "Function Key ";X;"
       : ";MID$(K$,1,J-1)
LI 260 PRINT:PRINT "Enter new de
       finition or press ENTER t
       o leave unchanged"
DB 270 LINE INPUT Q$:IF LEN(Q$)>
       0 THEN GOSUB 300:IF ER=1
       THEN ER=0:GOTO 250
LF 280 IF X+FIX(J/16)>9 THEN GOT
       O 380
JO 290 X=X+1+FIX(J/16):K=1619+16
       *(X-1)-1:J=1:CALL SCR!(ST
       K$):LOCATE 5,1:GOTO 200
DD 300 INPUT "Do you want a carr
       iage return (Y/N)";Q1$
GF 310 IF Q1$="Y" OR Q1$="y" THE
       N Q$=Q$+CHR$(13)
JJ 320 IF LEN(Q$)<16 THEN J=LEN(
       Q$):KEY X,Q$:RETURN
QK 330 M=1:N=16*(X-1)+1:IF N+LEN
       (Q$)>160 THEN BEEP:PRINT
       "Too long":ER=1:RETURN
PC 340 MID$(KN$,N,1)=MID$(Q$,M,1
       )
BF 350 M=M+1:N=1+N:IF M<=LEN(Q$)
       THEN 340
QN 360 IF LEN(Q$)>J THEN J=LEN(Q
       $)
NL 370 RETURN
PE 380 FOR X=1 TO 10
NP 390 IF ASC(MID$(KN$,16*(X-1)+
       1,1))>0 THEN FOR J=16*(X-
       1)+1 TO 16*X:POKE 1619+J-
       1,ASC(MID$(KN$,J,1)):NEXT
       J
FA 400 NEXT X:CLS:KEY LIST
JE 410 KB$=INKEY$:IF KB$="" THEN
       420 ELSE 410
IF 420 PRINT:INPUT "Do you want
       to save function keys as
       a BLOADable file (Y/N)";Q
       $
GJ 430 IF Q$="Y" OR Q$="y" THEN
       INPUT "Filename";F$ ELSE
       END
QM 440 BSAVE F$,1619,159:PRINT
EN 450 PRINT "To load your funct
       ion key file, use these c
       ommands:"
AA 460 PRINT:PRINT
NF 470 PRINT "DEF SEG:BLOAD ";CH
       R$(34);F$;CHR$(34);",1619
       :CLS":END
```

# Commodore 64 *SpeedScript* Fontmaker

Charles Brannon, Program Editor

*Special fonts add character to any screen display. This article shows how to use custom character sets with any version 3.0 or higher of Commodore 64* SpeedScript. *This month's premiere edition of the Commodore* COMPUTE! DISK *includes the Fontmaker programs and sample font listed here, plus version 3.2 of* SpeedScript.

Writing with a word processor often means staring for hours at a video screen. For word processing, screen clarity is especially vital. It's best to have a good-quality color or monochrome monitor, but a clear, readable character set helps, too. Commodore's built-in character set works well and is especially designed for the low resolution of the average TV. However, it can be improved. Besides, it's just plain fun to use your own custom character set. A custom font personalizes your computer and sets it apart from the crowd. There are many font editor programs to design character sets for use with BASIC, but until now there was no way to use them with *SpeedScript*.

Type in Program 1, "Fontmaker Boot," and save it to disk—preferably as the very first program on the directory (this lets you conveniently LOAD "*",8 to start the process). Program 1 configures memory for Program 2, "Fontmaker," which does the actual work. Fontmaker won't do its job unless you've run Program 1 first. *You must save Program 2 with the name FONTMAKER, since this is the name Program 1 looks for when it runs.* To use another name, change line 20 of Program 1.

Fontmaker only installs a character set that has been previously created; it has no provisions for creating the custom characters. You can easily define your own fonts or edit the supplied ones with a character editor such as "Ultrafont" (*COMPUTE!'s First Book of Commodore 64 Sound and Graphics*). This article includes one sample character set that you can type in. Also, this month's premiere of the COMPUTE! DISK includes *SpeedScript* 3.2, Fontmaker Boot, Fontmaker, and the sample font listed below.

When you run Fontmaker, it prompts you for the name of the character set you'd like to use. By default, the cursor blinks on the filename SPEED.SET. If you'd like to use a font with that name, just press RETURN. Otherwise, type in a new name, overwriting SPEED.SET. If you want to run *SpeedScript* without a custom set, just type X at the prompt (you don't need to erase SPEED.SET; just enter an X).

The character set you've previously created with a font editor program must be on the same disk as the *SpeedScript* program. Fontmaker looks for *SpeedScript* under the filename SS. Either insert a different filename in line 140 of Program 2 or rename your copy of *SpeedScript* to SS. Fontmaker loads in *SpeedScript*, bumps up the start of text space (reducing available memory by about 11K), loads the character set into that gap, switches the screen to the new character set, then runs *SpeedScript*.

## It's Only Temporary

Fontmaker does not permanently change *SpeedScript* unless you resave the word processor at this point (not recommended). In other words, Fontmaker installs the custom character set only for the current session. If you exit *SpeedScript* by pressing the RESTORE key, type POKE 53272,26 to restore the set before you type RUN to reenter *SpeedScript*.

When designing your custom character set, remember that vertical lines appear thinner and fuzzier than horizontal lines. Notice that every vertical line is doubled on the normal Commodore character set, making characters appear bold. You'll probably want to follow the same rule when designing your own sets. This is not a problem with crisp monochrome monitors. You can use the full 8 × 8 resolution of the character grid to design clean, well-formed characters.

Another guideline for readability is that uppercase characters are of uniform height. All lowercase characters are the same height, except for tall characters such as b, d, f, h, i, k, l, and t, which are the same height as uppercase letters. Normally you'll keep the rightmost column and the lowest row blank to keep characters from running into each other and to reserve room for

the lowercase descenders on the g, j, p, q, and y. Naturally, an exception is when you design cursive or script characters that should link together.

You'll also want to customize the punctuation marks and symbols. *SpeedScript* uses the back-arrow symbol as the carriage-return mark. If you don't like to see return-marks, just blank out that character. You can put a tiny dot in the SHIFT-SPACE character to distinguish it from a real space. It can also be convenient to define some of the graphics characters to their printing equivalent on the printer. For example, some graphics characters print as italic or foreign-language characters. Just edit the graphics characters to look like their printing equivalents.

You can also create your own custom cursor. *SpeedScript*'s cursor just alternates between the normal and reverse-video version of whatever character it's sitting on. The last 128 characters of a character set are the reverse-video ones. If you want an underline cursor, just copy the normal set down to the reverse-video area and draw a line through the bottom row of every character. Special characters can even have a unique cursor.

## A Free Sample

The final listing below is a sample character set for you to type in. To do this, you must use our machine language entry program "MLX." Be sure you read and understand the instructions for using MLX before you begin entering the data. (In case you missed its introduction last month, COMPUTE! now has an enhanced version of MLX. See the article for details.) Unlike most listings you enter with MLX, *this listing is not a machine language program*—it's pure character definition data. However, that fact doesn't matter to MLX, nor does it affect the way MLX operates. MLX still asks you for starting and ending addresses. For the character data, here are the proper values:

**Starting address: 7000**
**Ending address: 77FF**

When you finish entering the character set data, be sure to save a copy on the same disk with Fontmaker, Fontmaker Boot, and *SpeedScript*. If you wish this to be the default character set for the Fontmaker program, save the character data with the filename SPEED.SET. This is the default name used in Program 2 (line 170).

For instructions on entering these listings, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

## Program 1: Fontmaker Boot

```
10 PRINT"{CLR}{3 DOWN}POKE44,4
   8:POKE12288,0:NEW"   :rem 21
20 PRINT"{2 DOWN}LOAD"CHR$(34)
   "FONTMAKER"CHR$(34)",8"
                        :rem 122
30 PRINT"{4 DOWN}RUN{HOME}":PO
   KE198,3:POKE631,13:POKE632,
   13:POKE633,13:END   :rem 183
```

## Program 2: Fontmaker

```
100 REM DO NOT RUN THIS PROGRA
    M UNTIL YOU SAVE IT
                        :rem 147
110 POKE53280,6:POKE53281,14:S
    P$="{RVS}{40 @}"    :rem 86
120 PRINT"{CLR}{N}{4 DOWN}
    {YEL}"SP$"{RED}"SP$"{PUR}"
    SP$"{CYN}"SP$:PRINTTAB(8)"
    {WHT}{2 DOWN}LOADING SPEED
    SCRIPT..."          :rem 6
130 PRINT"{5 DOWN}{CYN}"SP$"
    {PUR}"SP$"{RED}"SP$"{YEL}"
    SP$                 :rem 151
140 F$="SS":ADR=2049:GOSUB210
                        :rem 144
150 F$="":PRINT"{BLK}{11 UP}WH
    ICH CHARACTER SET WOULD YO
    U LIKE?"            :rem 175
160 PRINT"(ENTER X FOR ROM SET
    )"                  :rem 80
170 PRINT"{23}? SPEED.SET
    {11 LEFT}";:INPUT F$:IFLEF
    T$(F$,1)="X"THEN190
                        :rem 200
180 ADR=10240:GOSUB210:POKE532
    72,26:POKE 2473,48:rem 194
190 POKE44,8:SYS2061    :rem 151
200 END                 :rem 105
210 OPEN1,8,0,F$        :rem 76
220 POKE780,1:POKE781,8:POKE78
    2,0:SYS65466        :rem 214
230 POKE780,0:POKE782,ADR/256:
    POKE781,ADR-PEEK(782)*256:
    SYS65493            :rem 243
240 CLOSE1:RETURN       :rem 87
```

## Sample Character Set

*The character set data must be entered with the MLX machine language entry program elsewhere in this issue. Refer to the MLX article before entering this listing.*

```
7000:7C C6 DE DE C0 C0 78 00 94
7008:00 00 78 0C 7C CC 76 00 BC
7010:E0 60 60 7C 66 66 FC 00 14
7018:00 00 3C 66 60 66 3C 00 FB
7020:0E 06 06 3E 66 66 3F 00 79
7028:00 00 3C 66 7E 60 3E 00 E8
7030:1C 36 30 78 30 30 78 00 6D
7038:00 00 76 CC CC 7C 0C F8 1E
7040:E0 60 6C 76 66 66 E6 00 39
7048:18 00 38 18 18 18 3C 00 57
7050:0C 00 0C 0C 0C CC CC 78 1F
7058:E0 60 66 6C 78 6C E6 00 98
7060:70 30 30 30 30 30 78 00 C1
7068:00 00 CC FE D6 C6 C6 00 32
7070:00 00 7C 66 66 66 66 00 E0
7078:00 00 3C 66 66 66 3C 00 8C
7080:00 00 DC 66 66 7C 60 F0 3A
7088:00 00 76 CC CC 7C 0C 1E 93
7090:00 00 DC 76 60 60 F0 00 DA
7098:00 00 7C C0 7C 06 FC 00 0B
70A0:10 30 7C 30 30 36 1C 00 BA
70A8:00 00 CC CC CC CC 7E 00 86
70B0:00 00 66 66 66 3C 18 00 19
70B8:00 00 C6 D6 FE FE 6C 00 AC
70C0:00 00 C6 6C 38 6C C6 00 42
70C8:00 00 66 66 66 3E 06 7C 91
70D0:00 00 7E 4C 18 32 7E 00 CC
70D8:7C 60 60 60 60 60 7C 00 9F
70E0:0C 12 30 7C 30 62 FC 00 1F
70E8:3E 06 06 06 06 06 3E 00 50
70F0:00 10 38 7C 10 10 10 00 85
70F8:00 00 0C 0C 58 60 70 00 41
7100:00 00 00 00 00 00 00 00 E2
7108:18 3C 3C 18 18 00 18 00 FF
7110:66 66 66 24 00 00 00 00 CE
7118:6C 6C FE 6C FE 6C 6C 00 75
7120:18 3E 60 3C 06 7C 18 00 C0
7128:00 C6 CC 18 30 66 C6 00 80
7130:38 6C 38 76 CC CC 76 00 3F
7138:0C 0C 18 00 00 00 00 00 27
7140:0C 18 30 30 30 18 0C 00 32
7148:30 18 0C 0C 0C 18 30 00 AC
7150:00 66 3C FF 3C 66 00 00 CF
7158:00 18 18 7E 18 18 00 00 4D
7160:00 00 00 00 00 18 18 30 04
7168:00 00 00 7E 00 00 00 00 33
7170:00 00 00 00 00 18 18 00 E3
7178:00 06 0C 18 30 60 C0 00 64
7180:7C C6 CE DE F6 C6 7C 00 E6
7188:18 38 18 18 18 18 7E 00 28
7190:3C 66 06 1C 30 66 7E 00 C5
7198:3C 66 06 1C 06 66 3C 00 F7
71A0:1C 3C 6C CC FE 0C 0C 00 3B
71A8:7E 60 7C 06 06 66 3C 00 15
71B0:1C 30 60 7C 66 66 3C 00 C6
71B8:7E 66 06 0C 18 18 18 00 47
71C0:3C 66 66 3C 66 66 3C 00 31
71C8:3C 66 66 3E 06 0C 38 00 E4
71D0:00 18 18 00 18 18 00 00 DD
71D8:00 18 18 00 00 18 18 30 85
71E0:0E 18 30 60 30 18 0E 00 DA
71E8:00 00 7E 00 7E 00 00 00 8F
71F0:70 18 0C 06 0C 18 70 00 95
71F8:3C 66 06 0C 18 00 18 00 06
7200:30 30 18 00 00 00 00 00 0C
7208:18 3C 66 66 7E 66 66 00 95
7210:FC 66 66 7C 66 66 FC 00 68
7218:3C 66 C0 C0 C0 66 3C 00 F0
7220:F8 6C 66 66 66 6C F8 00 A6
7228:FE 62 68 78 68 62 FE 00 84
7230:FE 62 68 78 68 60 F0 00 68
7238:3C 66 C0 C0 CE 66 3E 00 85
7240:C6 C6 C6 FE C6 C6 C6 00 E1
7248:3C 18 18 18 18 18 3C 00 6F
7250:1E 0C 0C 0C CC CC 78 00 14
7258:E6 66 6C 78 6C 66 E6 00 2A
7260:F0 60 60 60 62 66 FE 00 92
7268:C6 EE FE FE D6 C6 C6 00 9B
7270:C6 E6 F6 DE CE C6 C6 00 5E
7278:38 6C C6 C6 C6 6C 38 00 32
7280:FC 66 66 7C 60 60 F0 00 78
7288:78 CC CC CC DC 78 1C 00 44
7290:FC 66 66 7C 6C 66 F6 00 0D
7298:3C 66 70 38 0E 66 3C 00 49
72A0:7E 5A 18 18 18 18 3C 00 79
72A8:C6 C6 C6 C6 C6 C6 7C 00 32
72B0:C6 C6 C6 C6 C6 7C 38 00 88
72B8:C6 C6 C6 D6 FE EE C6 00 3A
72C0:C6 C6 6C 38 38 6C C6 00 CC
72C8:66 66 66 3C 18 18 3C 00 A4
72D0:FE C6 8C 18 32 66 FE 00 23
```

```
72D8:1C 30 30 60 30 30 1C 00 5E
72E0:18 18 18 00 18 18 18 00 2C
72E8:38 0C 0C 06 0C 0C 38 00 CF
72F0:00 00 03 3E 76 36 36 00 13
72F8:00 6C 8A 8C 8A 8A 6C 00 6A
7300:00 00 00 10 00 00 00 00 E7
7308:F0 F0 F0 F0 F0 F0 F0 F0 EE
7310:00 00 00 00 FF FF FF FF F6
7318:FF 00 00 00 00 00 00 00 FE
7320:00 00 00 00 00 00 00 1F 26
7328:C0 C0 C0 C0 C0 C0 C0 C0 0F
7330:CC CC 33 33 CC CC 33 33 7D
7338:03 03 03 03 03 03 03 03 1F
7340:00 00 00 00 CC CC 33 33 5A
7348:FF FE FC F8 F0 E0 C0 80 2B
7350:03 03 03 03 03 03 03 03 37
7358:18 18 18 1F 1F 18 18 18 E7
7360:00 00 00 00 0F 0F 0F 0F 29
7368:18 18 18 1F 1F 00 00 00 4F
7370:00 00 00 F8 F8 18 18 18 57
7378:00 00 00 00 00 00 FF FF 5F
7380:00 00 00 1F 1F 18 18 18 FA
7388:18 18 18 FF FF 00 00 00 84
7390:00 00 00 FF FF 18 18 18 20
7398:18 18 18 F8 F8 18 18 18 94
73A0:C0 C0 C0 C0 C0 C0 C0 C0 87
73A8:E0 E0 E0 E0 E0 E0 E0 E0 8F
73B0:07 07 07 07 07 07 07 07 97
73B8:FF FF 00 00 00 00 00 00 9F
73C0:FF FF FF 00 00 00 00 00 A7
73C8:00 00 00 00 00 FF FF FF AF
73D0:03 03 03 03 03 03 FF FF AE
73D8:00 00 00 00 F0 F0 F0 F0 DD
73E0:0F 0F 0F 0F 00 00 00 00 E5
73E8:18 18 18 F8 F8 00 00 00 3C
73F0:F0 F0 F0 F0 00 00 00 00 B9
73F8:F0 F0 F0 F0 0F 0F 0F 0F A3
7400:82 39 21 21 3E 38 84 78 03
7408:00 7C 86 72 82 32 89 36 2E
7410:D0 50 5C 42 59 D9 82 FC 59
7418:00 3C 42 99 96 99 42 3C CD
7420:11 09 39 41 99 99 41 1E E2
7428:00 3C 42 99 81 9E 41 3E 49
7430:22 49 4E 84 48 48 84 3C 37
7438:00 76 89 32 32 02 F2 04 96
7440:10 90 92 89 99 99 19 66 0C
7448:24 18 44 24 24 24 42 7E C8
7450:12 0C 12 12 D2 92 B2 44 33
7458:10 97 99 92 84 92 19 E6 13
7460:78 88 48 48 48 48 84 78 1A
7468:00 CC 33 01 29 39 29 C6 42
7470:00 7C 82 99 99 99 99 66 2F
7478:00 3C 42 99 99 99 42 3C 46
7480:00 DC 22 99 99 82 98 04 8A
7488:00 77 89 32 32 82 32 21 C4
7490:00 DC 22 89 96 90 08 F0 85
7498:00 7C 82 3C 82 79 02 FC AF
74A0:28 4C 82 4C 4C 40 22 1C 89
74A8:00 CE 32 32 32 32 81 7E 8A
74B0:00 66 99 99 99 42 24 18 36
74B8:00 C7 29 28 00 00 93 6E D0
74C0:00 C7 29 92 44 92 29 C7 70
74C8:00 66 99 99 99 41 79 82 5F
74D0:00 7E 81 B2 24 49 81 7E 7C
74D8:82 9E 90 90 90 9E 82 FE C8
74E0:12 2D 4C 82 4C 99 82 7C 1A
74E8:41 79 09 09 09 79 41 FF 33
74F0:10 28 44 82 EE 28 28 38 3D
74F8:00 1E 1E DE FE F8 F0 F8 D1
7500:00 00 3C 24 24 24 3C 00 DE
7508:24 42 42 24 24 18 24 18 02
7510:99 99 99 DB 24 00 00 00 40
7518:92 92 01 92 01 92 92 6E 20
7520:24 42 9C 42 39 82 64 1C 1E
7528:C6 39 F2 24 4E 99 29 C6 57
7530:44 52 46 89 32 32 89 76 17
7538:1E 12 32 24 18 00 00 00 FF
7540:12 24 48 48 48 64 32 0E 11
7548:48 24 12 12 12 26 4C 70 F5
7550:66 18 C3 00 C3 18 66 00 38
7558:3C 24 E7 81 E7 24 3C 00 C7
7560:00 00 00 3C 24 24 24 48 51
7568:00 00 7E 81 7E 00 00 00 2F
7570:00 00 00 00 18 24 24 18 0D
7578:06 09 12 24 48 90 20 40 32
7580:82 39 31 21 09 39 82 7C E1
7588:24 44 24 24 24 66 81 7E 99
7590:42 99 79 22 4C 99 81 7E 9E
7598:42 99 79 22 39 99 82 7C 0E
75A0:22 42 92 32 01 F2 12 1E B8
75A8:81 9E 82 79 F9 99 42 3C DA
75B0:22 4C 9C 82 99 99 42 3C 6F
75B8:81 99 79 32 24 24 24 3C 53
75C0:42 99 99 42 99 99 42 3C 7E
75C8:42 99 99 41 39 32 44 78 16
75D0:3C 24 3C 00 3C 24 3C 00 55
75D8:3C 24 3C 00 3C 24 24 78 A5
75E0:11 26 4C 98 4C 26 11 0F 1D
75E8:00 7E 81 7E 81 7E 00 00 91
75F0:88 64 32 19 32 64 88 F0 36
75F8:42 99 59 12 24 18 24 18 99
7600:78 48 4C 24 18 00 00 00 C7
7608:24 42 99 99 81 99 99 66 70
7610:02 99 99 82 99 99 02 FC F3
7618:42 99 26 20 26 99 42 3C AB
7620:04 92 99 99 99 92 04 F8 98
7628:01 9D 96 84 96 9D 01 FE 44
7630:01 9D 96 84 94 98 08 F0 28
7638:41 99 3E 3E 31 99 41 3E 88
7640:66 99 99 81 99 99 99 66 DE
7648:42 24 24 24 24 24 42 3C 98
7650:21 12 12 92 12 32 84 78 98
7658:1F 91 92 84 92 99 19 E6 E7
7660:08 90 90 92 95 99 01 FE C4
7668:29 11 01 01 29 39 29 C6 A5
7670:24 14 0D 21 31 29 29 C6 6F
7678:44 92 39 39 39 92 44 38 BB
7680:02 99 99 82 9C 90 08 F0 58
7688:04 32 32 32 22 84 62 1C 71
7690:02 99 99 82 92 99 09 F6 44
7698:42 99 8E 46 71 99 C2 7C 37
76A0:81 A5 66 24 24 26 42 3C 41
76A8:29 29 29 29 29 39 82 7C DB
76B0:29 29 29 29 39 82 44 38 C8
76B8:29 29 39 28 00 11 29 C6 8B
76C0:29 29 92 44 44 92 29 C7 A9
76C8:99 99 99 42 24 24 42 3C B2
76D0:01 F9 12 24 4C 9F 01 FE 23
76D8:22 4C C8 90 C8 4C 22 1E E5
76E0:3C 24 24 3C 3C 24 24 3C 34
76E8:44 32 13 09 13 32 44 78 D9
76F0:00 03 3C C1 89 C9 49 36 7E
76F8:FF 93 75 73 75 75 93 FF 59
7700:FF FF FF EF FF FF FF FF ED
7708:0F 0F 0F 0F 0F 0F 0F 0F F6
7710:FF FF FF FF 00 00 00 00 FE
7718:00 FF FF FF FF FF FF FF 07
7720:FF FF FF FF FF FF FF 00 0F
7728:3F 3F 3F 3F 3F 3F 3F 3F 17
7730:33 33 CC CC 33 33 CC CC B8
7738:FC FC FC FC FC FC FC FC 27
7740:FF FF FF FF 33 33 CC CC FB
7748:00 01 03 07 0F 1F 3F 7F 3B
7750:FC FC FC FC FC FC FC FC 3F
7758:E7 E7 E7 E0 E0 E7 E7 E7 9E
7760:FF FF FF FF F0 F0 F0 F0 6D
7768:E7 E7 E7 E0 E0 FF FF FF 57
7770:FF FF FF 07 07 E7 E7 E7 5F
7778:FF FF FF FF FF FF 00 00 67
7780:FF FF FF E0 E0 E7 E7 E7 DB
7788:E7 E7 E7 00 00 FF FF FF 62
7790:FF FF FF 00 00 E7 E7 E7 D6
7798:E7 E7 E7 07 07 E7 E7 E7 72
77A0:3F 3F 3F 3F 3F 3F 3F 3F 8F
77A8:1F 1F 1F 1F 1F 1F 1F 1F 97
77B0:F8 F8 F8 F8 F8 F8 F8 F8 9F
77B8:00 00 FF FF FF FF FF FF A7
77C0:00 00 00 FF FF FF FF FF AF
77C8:FF FF FF FF FF 00 00 00 B7
77D0:FC FC FC FC FC FC 00 00 C8
77D8:FF FF FF FF 0F 0F 0F 0F A9
77E0:F0 F0 F0 F0 FF FF FF FF B1
77E8:E7 E7 E7 07 07 FF FF FF 6B
77F0:0F 0F 0F 0F FF FF FF FF FD
77F8:0F 0F 0F 0F F0 F0 F0 00 33
```

©

# Atari RESET Controller

Torben Pedersen

*Here is a short machine language routine that traps the Atari SYSTEM RESET button in any BASIC program. An example program shows how disks can be protected with a password system that ignores BREAK and RESET. The routine works on any 400/800, XL, or XE with a disk drive.*

A well-designed program should accept any input without crashing. This can be done to some extent by screening input and disabling the BREAK key. However, if a person happens to hit the Atari SYSTEM RESET button, the program abruptly halts. The solution to this problem is to disable RESET. Unfortunately, although BREAK can be turned off with only a couple of POKEs, the RESET button cannot be disabled. It can, however, be trapped—meaning that you can divert it from resetting the system to doing something else. But this job requires a machine language (ML) routine.

"Atari RESET Controller" lets you trap RESET in any BASIC program even if you don't know anything about machine language. Here are the steps to follow:

1. Type in and save Programs 1, 2, and 3.

2. Load and run Program 1. It prints six program lines—one of which

contains strange graphics characters—on the screen. The odd-looking string (ML$) actually contains the encoded ML routine. The lines are numbered from 60–110 so they'll fit into Program 2.

3. Without disturbing lines 60–110 on the screen, type NEW and press RETURN, then move the cursor over line 60 and press RETURN six times, entering lines 60–110 into memory.

4. LIST the lines to disk by typing LIST"D:TEMP" and pressing RETURN. This stores the lines in ASCII form so they can be merged later with Program 2.

5. Load Program 2 into memory, then type ENTER"D:TEMP" and press RETURN. This merges lines 60–110 back into memory without disturbing the rest of Program 2.

6. Resave Program 2 by typing SAVE"D:LOGON" and pressing RETURN. The program is saved to disk under the filename LOGON, and you have saved yourself the trouble of trying to type in the odd-looking string that contains the ML routine. Don't run Program 2 yet.

7. Load Program 3, insert the disk that contains the LOGON file, then run the program. Program 3 creates an AUTORUN.SYS file that automatically loads and runs LOGON whenever you boot the disk.

## What's The Password?

Now that the package is complete, reboot the system by turning the computer off and on. The AUTORUN.SYS file loads and runs the LOGON program without any further action on your part.

When LOGON begins, it disables BREAK, traps RESET, and asks for a password. Until you type the right password, there's no way to break out of the program or proceed any further. In this case we know the password is SECRET (see line 300 of Program 2). Once it identifies you as an authorized user, LOGON restores BREAK and RESET, permitting the computer to work normally again. At this point, it's very important to reset the system by pressing RESET. If you omit this step, you won't be able to use the disk drive.

To use LOGON for your own programs, replace SECRET in line 300 with a password of your own. After that's done, the disk is effectively protected from use by anyone who doesn't know your password. Of course, somebody can circumvent this security system by booting from another disk, but this method should be sufficient for many purposes.

You might also want to trap RESET in a program intended for young children, or in any situation where a reset would cause problems. The ML routine created by Program 1 is actually quite simple. It diverts the computer from its normal reset routine to the custom routine stored in ML$. When you press RESET, the custom routine changes the character color from white to blue (to conceal printing), then prints RUN followed by a carriage return. As a result, pressing RESET reruns the program in memory.

For instructions on entering these listings, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

## Program 1: Atari RESET Controller

```
BD 10 REM *** PROGRAM TO CRE
      ATE THE
CD 20 REM *** MACHINE LANGUA
      GE ROUTINE
GH 30 REM *** IN STRING FORM
EI 40 REM
LI 50 DIM ML$(65)
PA 60 PRINT "60 DIM ML$(65)"
MI 70 PRINT "70 ML$=";CHR$(3
      4);
BE 80 FOR I=1 TO 65
MG 90 READ A
OM 100 PRINT CHR$(A);
BK 110 NEXT I
NJ 120 PRINT CHR$(34)
AK 130 PRINT "80 ADDR=ADR(ML
      $)"
EP 140 PRINT "90 HIGH=INT(AD
      DR/256)"
FG 150 PRINT "100 LOW=ADDR-H
      IGH*256"
MD 160 PRINT "110 POKE 12,LO
      W:POKE 13,HIGH"
BJ 170 DATA 169,46,72,169,53
      ,72,169,50,72,169,148
      ,141,197,2,169,0,141,
      68,2,169,1,133,9
EA 180 DATA 173,48,2,133,203
      ,173,49,2,133,204,160
      ,4,177,203,133,205
OA 190 DATA 200,177,203,133,
      206,162,0,160,82,104,
      145,205,232
HP 200 DATA 200,224,3,208,24
      7,169,12,141,252,2,10
      8,250,191
```

## Program 2: Logon

```
LP 150 DIM PASSWORD$(25)
MO 160 OPEN #1,4,0,"K:"
KC 200 GRAPHICS 0:SETCOLOR 2
      ,0,0
JK 210 POSITION 2,5:PRINT "L
      OGON:";
AK 220 POKE 16,64:POKE 53774
      ,64:REM DISABLE THE B
      REAK KEY
BG 260 GET #1,CHAR
CG 270 IF CHAR=155 OR LEN(PA
      SSWORD$)>25 THEN GOTO
       300
FC 280 PASSWORD$(LEN(PASSWOR
      D$)+1)=CHR$(CHAR)
GM 290 GOTO 260
PP 300 IF PASSWORD$="SECRET"
       THEN GOTO 340
KM 310 PASSWORD$=""
GH 320 GOTO 180
JB 360 POKE 12,64:POKE 13,21
      :REM RESET VECTOR
EG 370 POKE 16,192:POKE 5377
      4,192:REM ENABLE BREA
      K KEY
BM 380 GRAPHICS 0
HD 390 END
```

## Program 3: AUTORUN.SYS Maker

```
MB 10 REM *** PROGRAM TO CRE
      ATE AN
OC 20 REM *** AUTORUN.SYS FI
      LE TO
MN 30 REM *** EXECUTE LOGON
      ON BOOT UP
EI 40 REM
PF 50 OPEN #2,8,0,"D:AUTORUN
      .SYS"
MM 60 PUT #2,255:PUT #2,255
PL 70 PUT #2,0:PUT #2,6
DL 80 PUT #2,69:PUT #2,6
BB 90 FOR I=1 TO 70
OD 100 READ A:PUT #2,A
BK 110 NEXT I
IN 120 PUT #2,226:PUT #2,2
IP 130 PUT #2,227:PUT #2,2
CJ 140 PUT #2,0:PUT #2,6
GB 150 CLOSE #2
DK 160 DATA 169,148,141,197,
      2,169,0,141,68,2,169,
      1,133,9
DP 170 DATA 173,48,2,133,203
      ,173,49,2,133,204,160
      ,4,177,203,133,205
OH 180 DATA 200,177,203,133,
      206,162,0,160,82,189,
      58,6,145,205,232
LF 190 DATA 200,224,12,208,2
      45,169,12,141,252,2,1
      08,250,191
BA 200 DATA 50,53,46,2,36,26
      ,44,47,39,47,46,2
```

# Moving Marquee For Commodore 64

David W. Martin

*Have you ever seen commercial software that scrolls a message across the screen? Here is a short routine you can add to any BASIC program to achieve the same effect.*

How many times have you stared at the message PRESS ANY KEY TO CONTINUE? After using your computer for a while, you may become a bit tired of the same old screen displays. "Moving Marquee" lets you scroll any text message sideways across the top of the screen. Type in the program and save a copy, then run it to see how the marquee works.

Line 10 calls a subroutine at line 30000 which puts a machine language routine in memory. This needs to be done only once, when your program is performing setup tasks. Line 20 clears the top line of the screen and sets the corresponding area of color memory to white. Of course, you can use whatever color you like: To change the character color to red, change the 1 in line 20 to 2, and so on.

Line 30 lets you input the message of your choice. You may create the string any way that you like (for instance, A$="MESSAGE"), and the name of the string variable is not critical. However, you must add CHR$(0) to the end of the string (line 40) so the marquee routine knows where the message ends. In addition, since the routine always displays the last-defined string, you must not define any other strings before calling the routine with SYS.

Once you call the routine, it scrolls the entire string across the screen from right to left. Since this is done as a background task during the computer's hardware interrupt, the marquee display does not slow down the rest of your BASIC program. You may change the scrolling speed by POKEing a value from 0 to 128 into location 866 (the normal value is 5).

## Moving Marquee

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
10 GOSUB30000:PRINTCHR$(147)CH
   R$(17)                 :rem 67
20 FORJ=39TO0STEP-1:POKE1024+J
   ,32:POKE55296+J,1:NEXT
                          :rem 158
30 INPUT"ENTER MESSAGE";A$
                          :rem 90
40 A$=A$+CHR$(0)          :rem 24
50 POKE1009,PEEK(PEEK(71)+PEEK
   (72)*256+2)            :rem 12
60 POKE1011,PEEK(PEEK(71)+PEEK
   (72)*256+1)            :rem 5
70 SYS1008:END            :rem 64
30000 FOR ADR=864 TO 1015:READ
      BYT:POKE ADR,BYT:NEXT:R
      ETURN               :rem 45
30010 DATA 40,0,5,49,234,15,5,
      40,160,1,185,0,4,153,255
      ,3,200,204,96,3     :rem 28
30020 DATA 208,244,32,161,3,20
      5,97,3,240,15,192,255,24
      0,11,200,140,101    :rem 82
30030 DATA 3,172,96,3,153,255,
      3,96,172,96,3,169,32,153
      ,255,3,238,103,3:rem 123
30040 DATA 173,103,3,205,96,3,
      176,48,96,172,101,3,177,
      251,41,191,96       :rem 234
30050 DATA 141,251,0,142,252,0
      ,169,0,141,103,3,141,101
      ,3,173,20,3,141     :rem 10
30060 DATA 99,3,173,21,3,141,1
      00,3,120,169,223,141,20,
      3,169,3,141,21,3    :rem 79
30070 DATA 88,96,120,173,99,3,
      141,20,3,173,100,3,141,2
      1,3,88,96,206       :rem 223
30080 DATA 102,3,16,9,32,104,3
      ,173,98,3,141,102,3,108,
      99,3,162,151,169:rem 105
30090 DATA 205,32,169,3,96,0,3
      2,32,32,32,0        :rem 0
```

©

# Line Deleter For Atari

Bryce Wray

*Here's a short, simple programming utility that quickly deletes any range of lines within an Atari BASIC program. It works on all 400/800, XL, and XE computers.*

If you do much BASIC programming, you've undoubtedly needed "Line Deleter" at one time or another. There are only two other ways to delete a range of lines in Atari BASIC: the slow, manual method of typing each line number and pressing RETURN; and the roundabout method of listing to disk or tape the blocks of lines you want to keep, typing NEW, and then reentering the blocks into memory. Both techniques are cumbersome.

Line Deleter offers a better way. It's a little seven-line BASIC routine that takes advantage of *forced-read mode*—the Atari's ability to read information straight off its screen without any human intervention whatsoever. When needed, Line Deleter can be loaded from disk or tape and executed with a single command. As long as your program uses line numbers less than 32760, Line Deleter won't erase any part of it when loaded into memory.

## Using Line Deleter

Follow these steps:

1. Type in the listing below and save it on disk or tape with the LIST command, not SAVE or CSAVE. That is, LIST"C:" for cassette or LIST"D:*filename.ext*" for disk.

2. When you're ready to use Line Deleter, load it by typing ENTER"C:" for cassette or ENTER"D: *filename.ext*" for disk.

3. Type GOTO 32761 and press RETURN.

4. Screen prompts ask for three numbers: the beginning line number of the segment to be deleted; the ending line number of the segment; and the intervals between the lines. For example, if your program is numbered by tens, specify ten as the interval. If the program isn't so neatly numbered, you'll need to specify a different interval—perhaps even one. There's no problem if some of the line numbers are nonexistent.

That's it. Just sit back and watch Line Deleter do its stuff. Although it contains no machine language, it's pretty quick—on my 800XL, I've timed it at faster than 3.5 lines per second.

If you're unfamiliar with how the forced-read mode works, don't be disturbed by the STOPPED AT LINE 32764 messages you'll see flashing on the screen. The STOP statement in that line merely keeps the forced-read mode from running amok. You'll also see a CONT statement flashing onscreen; it keeps the routine going.

When Line Deleter is finished, the screen settles down and the usual READY message appears. At this point, you can resume working or use Line Deleter to erase another block of lines in your program. Since Line Deleter is still in memory, you can start with step 3.

## Eliminating Interference

Line Deleter uses so little RAM that you may want to keep it in memory at all times while programming. If so, I recommend inserting this line to keep it from interfering with your own program:

```
32760 END
```

This makes absolutely sure that your program won't accidentally run into Line Deleter. However, if you're pushing your Atari's RAM to its limits, you'll want to delete Line Deleter itself. Unfortunately, Line Deleter can't be used for this purpose. If you try, it devours the beginning of its critical FOR-NEXT loop and grinds to a halt. You'll have to erase it using one of the old-fashioned ways described above.

One note of caution: If you're using revision A or revision B Atari BASIC, Line Deleter can trigger the Atari lockup bug. This bug, which has plagued Atari programmers for years, can strike whenever any part of a BASIC program (even a single character) is deleted. There's no practical way to predict when it's going to happen, and usually the only cure is to switch the computer off and back on again—erasing your program, of course. Line Deleter neither increases nor decreases the chances of being bitten by the lockup bug.

If you're unsure which version of BASIC you have, type PRINT PEEK(43234) and press RETURN. If the result is 162, you have revision A; if the result is 96, you have revision B; if the result is 234, you have revision C. The only version free from the lockup bug (and a few other bugs, as well) is revision C, which is built into the 130XE or available as a cartridge for earlier Atari computers.

## Line Deleter

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
KA 32761 GRAPHICS 0:POSITION
          4,1:? "LINE DELETE
         R":? :?
GD 32762 ? "BEGINNING line n
         umber";:INPUT TOPLN
         O:? "ENDING line nu
         mber";:INPUT BOTLNO
         :? "Intervals";:INP
         UT STPR
MA 32763 FOR LINENO=TOPLNO T
         O BOTLNO STEP STPR
PD 32764 ? CHR$(125):POSITIO
         N 2,4:? LINENO:? "C
         ONT":POSITION 2,0:P
         OKE 842,13:STOP
GD 32765 POKE 842,12
AM 32766 NEXT LINENO
OA 32767 END                  ©
```

# Music Hath Charms

Whenever I attend a computer trade show I always look to see which exhibits seem to draw the most people. Generally, companies displaying musical products attract the biggest crowds.

Human beings have a continuing love affair with music that probably started when the first human heard a bird chirping. Each generation develops its own musical tastes, but there is a common thread that runs throughout the life of each of us—we love music.

Given the captivating power of music, it's little wonder that those of us who work with personal computers should want to use them to help us create music of our own. As I recall, the first peripheral I added to my Commodore PET in 1978 was a small amplifier I wired to the serial port of the PIA chip. By running a bit pattern through this port at different speeds, I was able to create simple musical tones. As crude as the sounds were by today's standards, they were musical enough to make the computer play a few compositions.

I was reminded of this project a few weeks ago when I came across my old PET lying in a corner of a closet. If I could have found the issue of the *PET Gazette* that showed how the hookup worked I might have brought the system out again, but my computer music tools have improved a lot since then.

## The Ideal Music Interface

When I bought my Apple II in 1979, I played with the sounds I could program through the Apple's built-in speaker. While the sound quality wasn't any better than what I could get with my PET, the built-in speaker in the Apple II motivated software developers to create music programs for this computer.

By the time the Atari 400 and 800 computers came out, musical support was getting much better. Programmers now had four voices to play with, each with independent control of volume and timbre. Even with this improved capability, I wanted more. As I played with the Atari (and, later, the Commodore 64), I remember being excited and frustrated at the same time. I was excited because inexpensive personal computers were capable of generating complex sounds, and frustrated because the tone quality was not as good as I wanted and musical data could not be captured simply by playing it in.

Entering musical notes by typing is cumbersome, and using a joystick is not much of an improvement. To my way of thinking, the personal computer was a wonderful tool for musical expression, but it was missing a natural user interface. Custom keyboards like those from Alpha Syntauri were a step in the right direction, but their cost kept all but professional, or die-hard amateur, musicians from achieving first-rate sounds with their computers.

I moved away from creating music on my personal computers and became more interested in the low-cost synthesizers that were appearing from companies like Casio. While these instruments didn't have the capacity to save my performance or to let me edit and print out a score, they did provide a natural user interface—a piano keyboard—and provided very high quality sound.

## The MIDI Breakthrough

Improvements in this field over the last three years have been spectacular. Now, for less than the price of a printer, you can purchase a polyphonic synthesizer that with one press of a button can change from a sixteenth century harpsichord to a space-age tone that sounds like a cross between a Chinese gong and a perturbed elephant.

Synthesizers have extraordinary sound generation capabilities, but they don't have the editing and storage facilities of a personal computer. To bring electronic music to its logical fruition, it seemed that someone would have to find a way to connect synthesizers to computers. Several inventive developers worked on this problem, and the invention of the MIDI interface marked the coming of age for computer-based music systems. Through a high-speed serial port, the MIDI interface allows personal computers to control, and be controlled by, special models of synthesizers. Yamaha and Casio were among the first synthesizer manufacturers to jump on the MIDI bandwagon, and numerous other companies (like Lowrey, Baldwin, and Wurlitzer) have adopted this standard as well.

The inexpensive CZ-101 synthesizer from Casio is one of the most popular MIDI instruments to date. With the CZ-101 (reviewed in this issue), you can create an extraordinary collection of sounds and can save sound libraries on removable cartridges. I have had this synthesizer connected to my Commodore SX-64 through the Passport MIDI card for quite some time. I now enjoy the power and expressive qualities of electronic music without the frustration I had with earlier systems.

Of all the ways personal computers can help people express themselves, the marriage of computers and music may end up being among the most important. Each of us has a song in our hearts, but only a few of us can write music well enough to get this song on paper. Through the interface between the synthesizer and the personal computer, anyone can pick out melodies on a keyboard, see them appear on the display screen, and then edit and refine them until they are just the way we want them. ©

# The World Inside the Computer

Fred D'Ignazio, Associate Editor

## The Ultimate Personal Computer

As a result of my work on a new book, I think I have stumbled onto the ultimate personal computer. *It's a robot!*

I'm working on a science-fiction trilogy for children based on the popular computer game *Robot Odyssey I* from the Learning Company. It's about a 19-year-old boy named Homer Pierce who is kidnapped by robot miners and carried down into Robotropolis, a robot world deep beneath the surface of the earth.

In the year 2005, human beings are surrounded by dozens of intelligent, aware, communicating machines. These artificial minds make all their decisions based on a narrow, specialist (I call it a "little-picture") perspective of the world. None of the machines sees the world from a broader, human perspective.

On his odyssey, the hero, Homer, comes to believe that personal robots can dramatically improve this situation. Homer would like to see people's primary relationship with machines (and technology) be through a *robot friend*. The robot would be a perfect middleman. It deals with the human on a cognitive, logical, and intellectual level, but is also aware of the human's physical, emotional, psychological, ethical, and spiritual nature. And it tries to advise and respond to the human with all these elements in mind. (This makes it a *big-picture machine*.) Then the robot translates what the human wants into commands and requests for all the specialist *little-picture machines*.

The robot friend has a human-like body because the human body is the best-engineered device for general-purpose mobility, sensing, and manipulating the environment. The robot is mobile, therefore, *portable*. It has immense storage and processing capabilities, but is also a *computer terminal* (with a built-in video screen and keyboard) that links a human (through electronic, digital, microwave communications) to the gigantic network of messages, pictures, voices, information, and music which is broadcast and relayed by satellite around the globe. The robot is a personalized, customized interface between the human and this network.

Each robot is fine-tuned to mirror and respond to the needs of its human friend. It becomes that human's private, personal agent. But it is not merely a machine; it's a high-tech, twenty-first century *Man Friday*.

### The Primary Robot Friend

As the primary robot becomes more attuned to the needs, personality, and humanity of its human friend, it spreads this awareness to all the little-picture machines it deals with. The primary robot acts as the human's agent, representing the human in all the dimensions of his or her professional and personal life. The robot encourages the machines to personalize their response to the human accordingly.

Also, the robot searches the global network for items of interest to the human being. It keeps these items in storage in the human's personal database (its robot memory—onboard and offboard in a storage closet in the home) and relates the items in newspaper, magazine, or conversational format whenever appropriate. In fact, the personal robot is an excellent conversationalist because: (1) It is extremely interested in anything its human friend has to say, so it is a good listener; (2) It loves to talk about things the human friend is interested in; and (3) It is an inexhaustible source of useful information.

### Secondary Robot Friends

The primary robot friend can accompany the human in the car, around town, at the office, and at home. But there are times when this becomes inappropriate or too costly. For those occasions, the human has small *secondary robot friends* to carry around. These robots are usually *laptop* or *pocket robots* which communicate directly with the primary robot friend and act as terminals between the human and the primary robot.

For example, if a human has a business meeting, he may want to take a secondary robot to the meeting and place the robot on the desktop in front of him. The secondary robot acts as a notebook or tape recorder and records the meeting. The human wears a cranial implant, a speaker/microphone biochip which enables him to be in direct, silent communication with the little robot at all times. He can ask the robot questions, have the little robot check with the big robot for advice, information, facts, statistics, and so on. Their "conversation" can be a lot like a conversation a human has with himself—*stream of consciousness*. It can include requests for facts pertinent to the meeting, items for a shopping list, or reminders to take an allergy pill or pick up the kids after school.

The robot is helpful to the human, but it does not take over his thinking. It is merely another voice, another "friend" the human can turn to. It is not to be used as a replacement for the human's own mind, imagination, judgment, or conscience. The robot plays Jiminy Cricket to the human's Pinocchio. The friend never has the authority to make decisions for the human, only to offer information and advice.

What do you think of my idea for the ultimate personal computer? What kind of robot friend or personal computer would *you* like to have? Write me c/o COMPUTE!, P.O. Box 5406, Greensboro, NC 27403.©

# The Beginners Page

Tom R. Halfhill, Editor

## The Power Of Strings

Last issue we introduced the concept of string variables and briefly hinted at their power—that their ability to hold strings of characters can let your programs manipulate words and sentences instead of just numbers. Consider for a moment how many programs manipulate text in some way: text editors, word processors, database managers, telecommunications programs, educational software, adventure games, even spreadsheets to some extent. Because math isn't the only language humans use to communicate ideas and manipulate information, over the years we've devised ways to make computers handle our alphabets as well.

But keep in mind that digital computers are still number-crunchers at heart. The alphabetic characters which appear on their monitor screens are merely an illusion created for our convenience. Internally, computers see the whole universe in terms of numbers, and they're unaware of anything that can't be translated into numbers. We'll discover some implications of this as we explore the uses of strings in BASIC.

### Reducing Redundancy

Probably the simplest way to begin taking advantage of strings in your programs is to use them to save memory and reduce typing. When you assign a string of characters to a string variable (A$="HELLO"), the computer stores the string in a safe place in memory. The string variable is like a bookmarker that reminds the computer where it is keeping the string. From then on, whenever you include that string variable in a BASIC statement, the computer looks up the string of characters in memory and carries out your command. If you print the variable, the entire string appears on the screen.

For example, if there are screen messages that frequently appear in different parts of your program—such as "PRESS ANY KEY TO CONTINUE" or "SELECT NUMBER OF MENU CHOICE"—it's a waste of memory and time to repeatedly type them in as separate PRINT statements. Instead, assign them to string variables like this:

```
10 A$="PRESS ANY KEY TO
   CONTINUE"
20 B$="SELECT NUMBER OF MENU
   CHOICE"
```

and then print the appropriate variable when you need to display the message:

```
100 PRINT A$
```

Here's another example: You've probably seen programs which draw horizontal rows of asterisks or dashes across the screen to make decorative borders, or to separate the screen into different sections for menus and so forth. Obviously it would waste memory to draw these lines with literal PRINT statements, since each PRINT would have to be followed by 40 or 80 characters (depending on the width of your computer's screen display). A better way is to use a FOR-NEXT loop, such as FOR X=1 TO 40:PRINT "*";:NEXT X. But if your program draws these lines often, you might save even more memory by defining a string variable with asterisks or dashes and then just printing the variable whenever you need it. This also executes faster than a FOR-NEXT loop.

### Strings With INPUT

Substituting string variables for literal PRINT statements is useful, but you really begin appreciating the power of string variables when you use them as *variables*. Like numeric variables, string variables can be manipulated in dozens of ways.

For instance, with an INPUT statement you can allow the user to assign and reassign characters to a string variable as the program runs—something a literal string can never do. Here's the most common example:

```
10 PRINT "WHAT IS YOUR NAME";
20 INPUT N$
30 PRINT "HELLO, ";N$
40 GOTO 10
```

(Make sure you type the semicolons *outside* the quotation marks in lines 10 and 30, and include the space between the comma and closing quotes in line 30. On Atari computers, don't forget you must always dimension a string variable before its first reference—insert the statement DIM N$(50) with a line number less than 10.)

When you run this program, it prints the message in line 10 and then waits at line 20 until the user types some characters and presses RETURN or ENTER. When the computer detects that RETURN or ENTER is pressed, it assigns whatever characters were typed to the string variable N$. Then it continues to line 30 and prints the HELLO message followed by the characters in N$. Finally, the computer returns to line 10 and lets the user assign a completely new string of characters to N$.

Since the content of N$ is determined by the user, not predetermined by the programmer, this little program can be the basis for a branching routine which takes different actions depending on the user's response. And that, in turn, is the basis for a wide variety of programs which tailor themselves to user input: educational programs that ask a question and evaluate the answer, programs that offer options and accept yes or no choices, programs that request you to specify a filename before loading or saving a data file—just about every kind of program, in fact. We'll take a closer look at these techniques and others in next month's column. ©

# INSIGHT: Atari

Bill Wilkinson

## Do You Need A 16-Bit Computer?

There has been a disturbing trend in my reader mail for the last couple of months. On the one hand, more and more people are asking for help: Where can I find out how to work with player/missile graphics? How do I hook a model 2300 argon laser to an Atari's joystick ports and shoot down unfriendly flying saucers? (That's not as much an exaggeration of the original question as you may think.) At the same time, and all too often from the same people, I hear of grandiose plans to buy an Atari ST or an Amiga and make the world safe for computocracy. I hate to burst any bubbles, but let's reason together for a moment.

Over the past six years there have been at least 60 or 70 books published about the Atari 8-bit computers. Some are great, some are terrible, and most are at least adequate. True, most of these books are hard to find. Three years ago, the bookshelves had a handful of books about dozens of different kinds of computers. Now, instead, we find dozens of books about a handful of computers. Still, your bookstore can usually order what you need. And if it can't, try an Atari dealer. If that doesn't work, try one of the bigger mail order places that specializes in Atari.

Anyway, here's my point: If you think information about the 8-bit line is sparse, wait until you try to find out anything about the 16-bit machines! As I write this, the only book published so far is called *Presenting the Atari ST*. But don't expect to learn much from it that isn't in Atari's own somewhat skimpy (though attractive) manual. Yes, I have heard of additional books that are "in the works." But how long do you think it will be before there are 60 or 70 titles?

So I'm asking: "Why buy one of the new machines? Why not buy an 800XL or 130XE?" On the basis of price alone, the 8-bit machines win handily. Atari recently announced a special promotion: 130XE, 1050 disk drive, 1027 printer, *AtariWriter*, and DOS 2.5 for $399. Use your TV for the video, throw in a better programming language or business package and a game or two, and you're ready to enjoy computing for about five bills. Try to do the same thing with a 520ST, and you're going to spend about $1,300 to $1,400, presuming you want a color monitor. For an equivalent Amiga, add about $800. What does this extra money buy?

### Theory Versus Practice

In theory, the 16-bit machines should run programs 4 to 20 times faster than the 8-bit beasts. In truth, speed depends on the language and how well it is implemented. ST Logo is generally no faster than 8-bit Atari Logo. And for anything except possibly heavy math and intensive disk operations, neither Amiga's ABasiC nor ST BASIC are significantly (i.e., more than 25 percent or so) faster than OSS BASIC XE running on an XL or XE computer.

How about the theories that the new machines can run larger programs, display better graphics, use mouse control, and so on? As I write this, those are mostly just theories, waiting for people to write software and prove them. I have often told people contemplating the purchase of a computer that they should seek out a piece of software to fulfill their needs first, and only then ask what machine(s) it runs on. I cannot emphasize that advice enough for these new computers.

Does this mean that I think everyone should buy 8-bit machines and forget the new ones? Not at all! I simply question whether most people can benefit from their as-yet unrealized potential. And even when their power finally arrives, how many home users will need more than what they get with, say, a 130XE? Business, scientific, and other users may very well need the extra speed and power, but it's pretty hard to justify an extra $500 to $1,500 if all you do with your computer is write a few letters a month and balance your checkbook.

What about people who want to learn how to program? They are total novices on computers, but enthusiasm is a great emptier of the pocketbook. Aside from the fact that there are lots of books on learning how to program an 800XL or 130XE, and none on how to use an ST or Amiga, how hard is it to learn to program on these new wonder machines? Well, writing plain-vanilla BASIC programs without graphics is reasonably easy. But that's easy on the XL and XE machines, also. Simple graphics, with lines and colors? Easy on both kinds of machines. Moving objects? Now we are getting to where it depends on the language: very easy with Atari 8-bit Logo, BASIC XE, and Amiga ABasiC; nearly impossible for a beginner with Atari BASIC or ST BASIC.

I guess I've made my points. As for me, I am moving on to the 16-bit machines. I am ready to learn new and different things, such as how artificial intelligence programs work. Such as how to manipulate multiple screen windows when writing a business application in Pascal. Such as...well, you get the idea. But I still enjoy programming in BASIC. And I still have a library of dozens of programs (mostly public domain and therefore free, or nearly so) which I enjoy on my 130XE. So I won't abandon any of you soon. As for yourself, think hard and read a lot before you abandon your trusty 8-bitter. ©

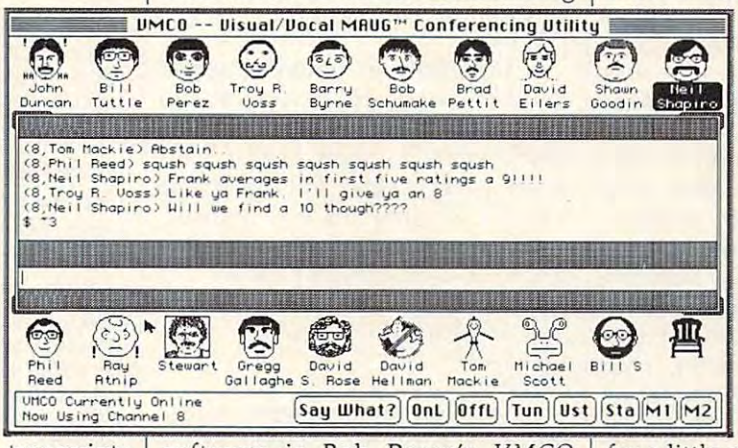
# Telecomputing Today

Arlan R. Levitan

## The Face Of Things To Come

Teleconferences via modem (COs) have been around on the various commercial information services for several years. Until recently, COs have typically looked something like this:

*(Arlan L.): What kind of computer are you folks using?*
*(Big Blue): I can't comment on that publicly.*
*(Jack T.): I'm using an Atari 2600 with Graduate keyboard...I've got a million of 'em.*
*(T. Leary): I don't need a computer...I'm plugged directly into the network.*




Pretty exciting, eh? Regular readers of this column are already aware that I am no great fan of participating in realtime teleconferencing. The complete transcripts of special "celebrity" COs are often available for perusal in the download sections. The complete text of a CO that went on for several hours can usually be downloaded in about ten minutes.

But now I have a confession to make. I participated in an online conference the other day and nearly split my sides in the process. Before you organize a lynch mob, let me explain.

The unique graphics and voice synthesis capabilities of Apple's Macintosh changed the face of teleconferencing via modem during the summer of 1985. Owners of modem-equipped Macintoshes can participate in conferences in which the faces of the participants appear on-screen, speak, and react to the other conferees.

Visual conferencing was spawned on the Delphi information service when Harry Chesley, a member of IconTact, Delphi's Macintosh SIG, set out to write (in his own words) "an insanely insane" program. Chesley wrote an interface between Apple's public domain *MacinTalk* speech synthesizer program and his own Visual Conference (VCO) telecommunications program. Visual/vocal conferencing was born.

The slickest and most recent incarnation of visual conferencing software is Bob Perez's *VMCO* (Visual/Vocal MAUG Conferencing Utility). *VMCO* was written for use on the conference section of CompuServe's Macintosh forum. While the basic function of Chesley's original is still intact, Perez has polished his implementation into a smooth, multifeatured program.

It's hard to describe the experience of a *VMCO* conference in words, although the phrase "organized lunacy" comes fairly close. The accompanying screen dump shows a 19-person conference in progress. All of the "chairs" in the "conference room" start out empty. As the conference starts to roll, *VMCO* checks the name of conference participants against the face files available on the disk from which *VMCO* was started. If a conferee's face file is found, it is "seated" in one of the chairs. If no face file is found, a generic face is seated instead. As the conferees type away at their keyboards, the words are "spoken" by their faces via *MacinTalk*, replete with lip movement. The conferees may have voices of different pitches and speeds and may also change their faces during the conference to indicate varying emotions.

Getting started with *VMCO* is not without its difficulties. The system is memory-intensive, requiring a 512K Mac. As of October 1985, the only way to design your own face is with a program called a "Resource Editor," which is beyond the experience of most casual Mac users. Then there's the veritable slew of files required: *VMCO*, *MacinTalk*, the Face Files created by other users, and three or four others. If you can't find a friend who already has *VMCO*, you're in for a little over two hours of downloading. Interested? If you're a CompuServe subscriber, the documentation can be found in the telecommunications download section of the MAUG Mac Forum (page PCS-23).

Will visual/vocal conferencing become available on other computers? That's hard to say. The Fat Mac's large memory, icons, and the speed of its Motorola 68000 central processing unit are what makes *VMCO* tick. I doubt that the eight-bit Commodore, Atari, and Apple computers have the oomph needed for visual conferencing. The IBM PC-AT has enough power, but when equipped with a suitable graphics adapter and display, you'll have spent more than eight thousand bucks. That leaves the Atari ST and Amiga as the most likely candidates for future visual conferencing. However, I'm not placing any bets at this time...if I'm wrong I might lose face. ©

# Programming the TI

C. Regena

## Music And Sound On The TI

Music and sound on the TI can be a lot of fun and fairly easy to program. Some computers require several statements to even play one note, but the TI can play an entire chord with one statement. The best way to learn to program music and sound is to sit at the console and experiment. This month we'll look at a few techniques.

The basic sound statement is CALL SOUND(d,f,v) where d is duration, f is frequency, and v is volume. You may specify more than one frequency and volume for each statement to hear more voices.

The duration parameter tells the computer how many milliseconds (thousandths of a second) the sound should last. CALL SOUND (1000,262,1) plays middle C for exactly one second. You can use this feature for any kind of timing, with or without sound. For example, by setting the volume to the softest and using a high frequency out of hearing range, a program can silently count off seconds.

In music programs it's helpful to use a variable for the duration. For example, let T represent a quarter note. T/2 will be an eighth note, T/3 a triplet, 2*T a half note, 4*T a whole note, and so on. Before the sound statements, define a value for T.

```
110 T=400
120 CALL SOUND(T,262,2)
130 CALL SOUND(T/2,294,2)
140 CALL SOUND(T/2,330,2)
150 CALL SOUND(2*T,349,2)
160 CALL SOUND(4*T,392,2)
170 END
```

To change the tempo, you won't need to change each sound statement, only line 110. For example, change set T=200, then RUN. The tempo changes with all the notes in proportion.

The TI can execute other statements, such as calculations or graphics, while making sounds. Last month's Christmas program is an example of graphics commands being executed among music commands. If another sound statement is encountered, the computer waits until the previous duration is finished. If you want the computer to execute a sound statement without waiting for the previous duration to finish, use a negative number for the duration:

```
110 CALL SOUND(2000,440,2)
120 CALL SOUND(-400,262,2)
130 END
```

The first note should be played for two seconds. However, line 120 includes a negative duration, so its sound starts as soon as the computer gets to line 120, and the sound continues for 400 milliseconds. Negative durations are often placed in a FOR-NEXT loop:

```
110 FOR F=262 TO 392 STEP 12
120 CALL SOUND(-200,F,2)
130 NEXT F
140 END
```

To determine frequency values for notes, consult the charts in the manuals that came with your computer. You can use these charts to translate sheet music. For example, CALL SOUND(1000,440,2) plays A at concert pitch. To play a chord, you can list three frequencies and volumes with one duration in a statement:

```
CALL SOUND(1200,262,2,330,2,392,2)
```

But you're not limited to numbers on the chart. For example, the frequency for middle C is 262, and the frequency for D is 294. You can play any tone between these notes:

```
110 FOR F=262 to 294
120 CALL SOUND(300,F,2)
130 PRINT F
140 NEXT F
150 END
```

By varying the frequency in a FOR-NEXT loop, you can create interesting sound effects:

```
110 FOR F=440 TO 523 STEP 15
120 CALL SOUND(-100,F,2)
130 NEXT F
140 FOR F=262 TO 131 STEP -10
150 CALL SOUND(-100,F,2)
160 NEXT F
170 END
```

Create noises by using negative frequencies from -1 to -8. These noises can be fun to add to games. However, you're not limited to just these noises. You may combine up to three other frequencies with one noise—you can spend days experimenting with different combinations to make different noises. Try these examples:

```
CALL SOUND(1000,-6,2,440,2)
CALL SOUND(1000,-6,2,262,2)
CALL SOUND(1000,-6,2,131,2,165,2)
```

The volume parameter may be a value from 0 (loudest) to 30 (softest). You can assign different volumes to notes to create dynamics, such as a crescendo, or to make a melody more prominent.

```
110 CALL SOUND(400,262,8)
120 CALL SOUND(400,294,6)
130 CALL SOUND(400,330,4)
140 CALL SOUND(400,349,2)
150 CALL SOUND(800,392,0)
160 END
```

Try varying the volume in loops to create sound effects:

```
110 FOR V=0 TO 30
120 CALL SOUND(-100,262,V)
130 NEXT V
140 FOR V=30 TO 0 STEP -1
150 CALL SOUND(-100,-5,V)
160 NEXT V
170 FOR V=0 TO 30
180 CALL SOUND(-100,-6,V)
190 NEXT V
200 FOR F=262 TO 330 STEP 34
210 FOR V=0 TO 30
220 CALL SOUND(-100,F,V,-6,V)
230 NEXT V
240 NEXT F
250 END
```

CALL SOUND is quite versatile and can add a lot to your programs. Take the time to experiment and you'll discover that you can create all kinds of sounds with your TI. ©

# IBM Personal Computing

Donald B. Trivette

## Last Minute Gifts

This is the first column of 1986 and a good place to tell you about three products for the IBM PC and PCjr that didn't fit into last year's columns, but which make great gifts.

Realia (pronounced Ree-AL-ia) has a program called *SpaceMaker* that actually compresses the size of programs so you can get more on a disk. *SpaceMaker* reduces the size of *most* program files—those ending in .COM or .EXE—but cannot compress data files. For example, it reduces the size of the IBM spelling checker *Word Proof*, but cannot reduce the size of *Word Proof*'s dictionary file.

At the technical level, *SpaceMaker* removes all the binary zeros from a program and writes them in a compact form. It then appends a little-bitty (pun intended) program to the beginning of the file. When DOS loads the file—when you type the name of the program—the tiny preprogram takes control and reconstructs the binary zeros as it loads and runs the bigger program. All this happens automatically.

*SpaceMaker* is so simple to use that you don't have to know anything about binary zeros or programming. All you need do is enter the input filename and the output filename and *SpaceMaker* does the rest. It generates a new, smaller program file which works just like the old one; the output filename is the new program name. As always, it's best to keep the original copy of a program on one disk and the *SpaceMaker*-squeezed copy on another.

Here are some typical space savings:

| Program | Original size | New size | Reduction |
|---|---|---|---|
| *IBM Personal Editor* | 45,696 | 41,728 | 8% |
| *Word Proof* | 27,056 | 24,616 | 9% |
| *PC-Talk III* (compiled version) | 81,408 | 66,880 | 17% |
| *Lotus 1-2-3* (1A) | 89,984 | 80,000 | 11% |
| BASICA.COM | [Won't compress] | | |

*SpaceMaker* retails for $75 and is produced by Realia, Inc., 10 South Riverside Plaza, Chicago, IL, 60606. It requires a PC or PCjr with a disk drive.

### A Hidden Typewriter

Even if you dislike desk-management software as much as I do, you might like *ProType*. It hides in memory like desk-management software until you need it, then is brought forth by pressing the ALT key twice. A *1-2-3*-like menu appears at the top of the screen. Selecting the Type option puts the program in typewriter mode. Any line typed on the computer goes to the printer. This is the mode I use most of the time: it's perfect for addressing envelopes and mailing labels. Pressing the ESC key sends *ProType* back into memory, where it occupies about 28,000 bytes, and returns you to your regularly scheduled program. I can whip out an envelope from the middle of *1-2-3* in 15 seconds!

**Even if you dislike desk-management software as much as I do, you might like *ProType*.**

But there's more to *ProType* than type mode. You can enter edit mode and compose and print (but not save) a one-page memo or letter. You can also create a template and use *ProType* to type forms, such as invoices, statements, and checks. Another command sends escape codes to the printer.

I'm amazed that it works happily with all the other things I have hiding in memory, namely a print spooler, a screen-blank-after-five-minutes program, a RAM disk, a keyboard enhancer, and a disk drive analyzer.

*ProType* retails for $69 and is from Photon Software, 14021 NE 8th Street, Bellevue, WA, 98009. It requires a PC or PCjr with a disk drive.

### Portable Sound

The third product is for PCjr owners who don't have a monitor with a built-in speaker and don't want to drag a stereo amp and hi-fi speakers to Junior's location. I'm in this group, so I've never been able to hear the wonderful sound effects, for example, in the *King's Quest* games. What I needed was an inexpensive amplifier-speaker that I could plug into the audio jack on Junior's backside.

I've found one. Radio Shack sells a battery-operated 200-milliwatt amplifier-speaker (catalog number 277-1008B) that's perfect and costs only $12. To hook it up to a PCjr, you'll need a cable (mini-phono to RCA plug) which costs about $2 at Radio Shack. ©

# COMPUTE!'s Guide To Typing In Programs

Before typing in any program, you should familiarize yourself with your computer. Learn how to use the keyboard to type in and correct BASIC programs. Read your manuals to understand how to save and load BASIC programs to and from your disk drive or cassette unit. Computers are precise—take special care to type the program *exactly* as listed, including any necessary punctuation and symbols, except for special characters as noted below. To help you with this task, we have implemented a special listing convention as well as a program to help check your typing—the "Automatic Proofreader." Please read the following notes before typing in any programs from COMPUTE!. They can save you a lot of time and trouble.

Commodore, Apple, and Atari programs can contain some hard-to-read (and hard-to-type) special characters, so we have developed a listing system that indicates the function of these control characters. (There are no special control characters in our IBM or TI-99/4A listings.) You will find Commodore and Atari special characters within curly braces; *do not type the braces.* For example, {CLEAR} or {CLR} instructs you to insert the symbol which clears the screen on the Atari or Commodore machines. For Commodore, Apple, and Atari, a symbol by itself within curly braces is usually a control key or graphics key. If you see {A}, hold down the CTRL key and press A. This will produce a reverse video character on the Commodore (in quote mode), a graphics character on the Atari, and an invisible control character on the Apple. Commodore computers also have a special control key labeled with the Commodore logo. Graphics characters entered with the Commodore logo key are enclosed in a special bracket that looks like this: [<A>]. In this case, you would hold down the Commodore logo key as you type A. Our Commodore listings are in uppercase, so shifted symbols are underlined. A graphics heart symbol (SHIFT-S) would be listed as S. One exception is {SHIFT-SPACE}. When you see this, hold down SHIFT and press the space bar. If a number precedes a symbol, such as {5 RIGHT}, {6 S}, or [<8 Q>], you would enter five cursor rights, six shifted S's, or eight Commodore-Q's. On the Atari, inverse characters (printed in white on black) should be entered after pressing the inverse video key.

Since spacing is sometimes important, any more than two spaces will be listed. For example, {6 SPACES} means to press the space bar six times. Our listings never leave a space at the end of a line, instead moving it to the next printed line as {SPACE}. For your convenience, we have prepared this quick-reference chart for the Commodore and Atari special characters:

## Atari 400/800/XL/XE

| When you see | Type | See | |
|---|---|---|---|
| {CLEAR} | ESC SHIFT < | | Clear Screen |
| {UP} | ESC CTRL - | | Cursor Up |
| {DOWN} | ESC CTRL = | | Cursor Down |
| {LEFT} | ESC CTRL + | | Cursor Left |
| {RIGHT} | ESC CTRL * | | Cursor Right |
| {BACK S} | ESC DELETE | | Backspace |
| {DELETE} | ESC CTRL DELETE | | Delete character |
| {INSERT} | ESC CTRL INSERT | | Insert character |
| {DEL LINE} | ESC SHIFT DELETE | | Delete line |
| {INS LINE} | ESC SHIFT INSERT | | Insert line |
| {TAB} | ESC TAB | | TAB key |
| {CLR TAB} | ESC CTRL TAB | | Clear tab |
| {SET TAB} | ESC SHIFT TAB | | Set tab stop |
| {BELL} | ESC CTRL 2 | | Ring buzzer |
| {ESC} | ESC ESC | | ESCape key |

## Commodore PET/CBM/VIC/64/128/16/+4

| When You Read: | Press: | See: | When You Read: | Press: | See: |
|---|---|---|---|---|---|
| {CLR} | SHIFT + CLR/HOME | | [<1>] | COMMODORE + 1 | |
| {HOME} | CLR/HOME | | [<2>] | COMMODORE + 2 | |
| {UP} | SHIFT + ↑ CRSR ↓ | | [<3>] | COMMODORE + 3 | |
| {DOWN} | ↑ CRSR ↓ | | [<4>] | COMMODORE + 4 | |
| {LEFT} | SHIFT + ← CRSR → | | [<5>] | COMMODORE + 5 | |
| {RIGHT} | ← CRSR → | | [<6>] | COMMODORE + 6 | |
| {RVS} | CTRL + 9 | | [<7>] | COMMODORE + 7 | |
| {OFF} | CTRL + 0 | | [<8>] | COMMODORE + 8 | |
| {BLK} | CTRL + 1 | | {F1} | f1 | |
| {WHT} | CTRL + 2 | | {F2} | SHIFT + f1 | |
| {RED} | CTRL + 3 | | {F3} | f3 | |
| {CYN} | CTRL + 4 | | {F4} | SHIFT + f3 | |
| {PUR} | CTRL + 5 | | {F5} | f5 | |
| {GRN} | CTRL + 6 | | {F6} | SHIFT + f5 | |
| {BLU} | CTRL + 7 | | {F7} | f7 | |
| {YEL} | CTRL + 8 | | {F8} | SHIFT + f7 | |
| | | | ← | ← | |

## The Automatic Proofreader

We have developed a series of simple, yet effective programs that can help check your typing. Type in the appropriate Proofreader program listed below, then save it for future use. On the VIC, 64, or Atari, run the Proofreader to activate it, then enter NEW to erase the BASIC loader (the Proofreader remains active, hidden in memory, as a machine language program). Pressing RUN/STOP–RESTORE or SYSTEM RESET deactivates the Proofreader. You can use SYS 886 to reactivate the VIC/64 Proofreader, or PRINT USR(1536) to reenable the Atari Proofreader. On the Apple, the Proofreader automatically erases the BASIC portion of itself after you activate it by typing RUN, leaving only the machine language portion in memory. It works with either DOS 3.3 or ProDOS. Disable the Apple Proofreader by pressing CTRL-RESET before running another BASIC program. The IBM Proofreader is a BASIC program that simulates the IBM BASIC line editor, letting you enter, edit, list, save, and load programs that you type. Type RUN to activate.

Once the Proofreader is active, try typing in a line. As soon as you press RETURN, either a decimal number (on the Commodore), a hexadecimal number (on the Apple), or a pair of letters (on the Atari or IBM) appears. The number or pair of letters is called a *checksum*. Try making a change in the line, and notice how the checksum changes.

All you need to do is compare the value provided by the Proofreader with the checksum printed in the program listing in the magazine. In Commodore listings, the checksum is a number from 0 to 255. It is set off from the rest of the line with *rem*. This prevents a syntax error if the checksum is typed in, but the REM statements and checksums need *not* be typed in. It is just there for your information.

In Atari, Apple, and IBM listings, the checksum is given to the left of each line number. Just type in the program one line at a time (without the printed checksum) and compare the checksum generated by the Proofreader to the checksum in the listing. If they match, go on to the next line. If not, check your typing: You've made a mistake. On the Commodore, Atari, and Apple Proofreaders, spaces are not counted as part of the checksum, so be sure you type the right number of spaces between quote marks. The Commodore and Atari Proofreaders do not check to see that you've typed the characters in the right order, so if characters are transposed, the checksum still matches the listing. Because of the checksum method used, do not type abbreviations, such as ? for PRINT. The IBM Proofreader is the pickiest of all; it *will* detect errors in spacing and transposition. Be sure to leave Caps Lock on, except when typing lowercase characters.

## IBM Proofreader Commands

Since the IBM Proofreader replaces the computer's normal BASIC line editor, it has to include many of the direct-mode IBM BASIC commands. The syntax is identical to IBM BASIC. Commands simulated are LIST, LLIST, NEW, FILES, SAVE, and LOAD. When listing your program, press any key (except Ctrl-Break) to stop the listing. If you type NEW, the Proofreader prompts you to press Y to be sure you mean yes.

Two new commands are BASIC and CHECK. BASIC exits the Proofreader back to IBM BASIC, leaving the Proofreader in memory. CHECK works just like LIST, but shows the checksums along with the listing. After you have typed in a program, save it to disk. Then exit the Proofreader with the BASIC command, and load the program in BASIC as usual (this replaces the Proofreader in memory). You can now run the program, but you may want to resave it to disk. The version of your program that you resave from BASIC will take up less space on disk and will load faster, but it can no longer be edited with the Proofreader. If you want to convert a program to Proofreader format, save it to disk with SAVE *"filename"*,A.

## Special Proofreader Notes For Commodore Cassette Users

The Proofreader resides in a section of memory called the cassette buffer, which is used during tape LOADs and SAVEs. Therefore, be sure to press RUN/STOP–RESTORE to get the Proofreader out of the way before saving or loading a program. If you want to use the Proofreader with tape, run the Proofreader, then enter these two lines *exactly* as shown, pressing RETURN after each one:

```
A$="PROOFREADER.T":B$="{10
 SPACES}":FOR X=1 TO 4:A$=A$
 +B$:NEXT
 FOR X=886 TO 1018:A$=A$+CHR$
(PEEK(X)):NEXT:OPEN 1,1,1,A$:
 CLOSE1
```

Then insert a blank tape and press RECORD and PLAY to save a special version of the Proofreader. Anytime you need to reload the Proofreader after it has been erased—for example, after you reload a paritally completed program—just rewind the tape, type OPEN1:CLOSE1, then press PLAY. You'll see the message FOUND PROOFREADER.T, but not the familiar LOADING message. Don't worry; the Proofreader is in memory. When READY comes back, enter SYS 886.

## Program 1: VIC/64 Proofreader

*By Charles Brannon, Program Editor*

```
10 PRINT"{CLR}PLEASE WAIT...":
   FORI=886TO1018:READA:CK=CK+
   A:POKEI,A:NEXT
20 IF CK<>17539 THEN PRINT"
   {DOWN}YOU MADE AN ERROR":PR
   INT"IN DATA STATEMENTS.":EN
   D
30 SYS886:PRINT"{CLR}{2 DOWN}P
   ROOFREADER ACTIVATED.":NEW
40 DATA 173,036,003,201,150,20
   8,001,096,141,151,003,173
50 DATA 037,003,141,152,003,16
   9,150,141,036,003,169,003
60 DATA 141,037,003,169,000,13
   3,254,096,032,087,241,133
70 DATA 251,134,252,132,253,00
   8,201,013,240,017,201,032
80 DATA 240,005,024,101,254,13
   3,254,165,251,166,252,164
90 DATA 253,040,096,169,013,03
   2,210,255,165,214,141,251
100 DATA 003,206,251,003,169,0
    00,133,216,169,019,032,210
110 DATA 255,169,018,032,210,2
    55,169,58,032,210,255,166
120 DATA 254,169,000,133,254,1
    72,151,003,192,087,208,006
130 DATA 032,205,189,076,235,0
    03,032,205,221,169,032,032
140 DATA 210,255,032,210,255,1
    73,251,003,133,214,076,173
150 DATA 003
```

## Program 2: Atari Proofreader

*By Charles Brannon, Program Editor*

```
100 GRAPHICS 0
110 FOR I=1536 TO 1700:RE
    AD A:POKE I,A:CK=CK+A
    :NEXT I
120 IF CK<>19072 THEN ? "
    Error in DATA Stateme
    nts.  Check Typing.":
    END
130 A=USR(1536)
140 ? :? "Automatic Proof
    reader Now Activated.
    "
150 END
160 DATA 104,160,0,185,26
    ,3,201,69,240,7
170 DATA 200,200,192,34,2
    08,243,96,200,169,74
180 DATA 153,26,3,200,169
    ,6,153,26,3,162
190 DATA 0,189,0,228,157,
    74,6,232,224,16
200 DATA 208,245,169,93,1
    41,78,6,169,6,141
210 DATA 79,6,24,173,4,22
    8,105,1,141,95
```

```
220 DATA 6,173,5,228,105,
    0,141,96,6,169
230 DATA 0,133,203,96,247
    ,238,125,241,93,6
240 DATA 244,241,115,241,
    124,241,76,205,238
250 DATA 0,0,0,0,0,32,62,
    246,8,201
260 DATA 155,240,13,201,3
    2,240,7,72,24,101
270 DATA 203,133,203,104,
    40,96,72,152,72,138
280 DATA 72,160,0,169,128
    ,145,88,200,192,40
290 DATA 208,249,165,203,
    74,74,74,74,24,105
300 DATA 161,160,3,145,88
    ,165,203,41,15,24
310 DATA 105,161,200,145,
    88,169,0,133,203,104
320 DATA 170,104,168,104,
    40,96
```

## Program 3: IBM Proofreader

*By Charles Brannon, Program Editor*

```
10 'Automatic Proofreader Ver
   sion 2.00 (Lines 270,510,5
   15,517,620,630 changed fro
   m V1.0)
100 DIM L$(500),LNUM(500):COL
    OR 0,7,7:KEY OFF:CLS:MAX=
    0:LNUM(0)=65536!
110 ON ERROR GOTO 120:KEY 15,
    CHR$(4)+CHR$(70):ON KEY(1
    5) GOSUB 640:KEY (15) ON:
    GOTO 130
120 RESUME 130
130 DEF SEG=&H40:W=PEEK(&H4A)
140 ON ERROR GOTO 650:PRINT:P
    RINT"Proofreader Ready."
150 LINE INPUT L$:Y=CSRLIN-IN
    T(LEN(L$)/W)-1:LOCATE Y,1
160 DEF SEG=0:POKE 1050,30:PO
    KE 1052,34:POKE 1054,0:PO
    KE 1055,79:POKE 1056,13:P
    OKE 1057,28:LINE INPUT L$
    :DEF SEG:IF L$="" THEN 15
    0
170 IF LEFT$(L$,1)=" " THEN L
    $=MID$(L$,2):GOTO 170
180 IF VAL(LEFT$(L$,2))=0 AND
     MID$(L$,3,1)=" " THEN L$
    =MID$(L$,4)
190 LNUM=VAL(L$):TEXT$=MID$(L
    $,LEN(STR$(LNUM))+1)
200 IF ASC(L$)>57 THEN 260 'n
    o line number, therefore
    command
210 IF TEXT$="" THEN GOSUB 54
    0:IF LNUM=LNUM(P) THEN GO
    SUB 560:GOTO 150 ELSE 150
220 CKSUM=0:FOR I=1 TO LEN(L$
    ):CKSUM=(CKSUM+ASC(MID$(L
    $,I))*I) AND 255:NEXT:LOC
    ATE Y,1:PRINT CHR$(65+CKS
    UM/16)+CHR$(65+(CKSUM AND
     15))+" "+L$
230 GOSUB 540:IF LNUM(P)=LNUM
     THEN L$(P)=TEXT$:GOTO 15
    0 'replace line
240 GOSUB 580:GOTO 150 'inser
    t the line
260 TEXT$="":FOR I=1 TO LEN(L
    $):A=ASC(MID$(L$,I)):TEXT
    $=TEXT$+CHR$(A+32*(A>96 A
    ND A<123)):NEXT
270 DELIMITER=INSTR(TEXT$," "
    ):COMMAND$=TEXT$:ARG$="":
    IF DELIMITER THEN COMMAND
    $=LEFT$(TEXT$,DELIMITER-1
    ):ARG$=MID$(TEXT$,DELIMIT
    ER+1) ELSE DELIMITER=INST
    R(TEXT$,CHR$(34)):IF DELI
    MITER THEN COMMAND$=LEFT$
    (TEXT$,DELIMITER-1):ARG$=
    MID$(TEXT$,DELIMITER)
280 IF COMMAND$<>"LIST" THEN
    410
290 OPEN "scrn:" FOR OUTPUT A
    S #1
300 IF ARG$="" THEN FIRST=0:P
    =MAX-1:GOTO 340
310 DELIMITER=INSTR(ARG$,"-")
    :IF DELIMITER=0 THEN LNUM
    =VAL(ARG$):GOSUB 540:FIRS
    T=P:GOTO 340
320 FIRST=VAL(LEFT$(ARG$,DELI
    MITER)):LAST=VAL(MID$(ARG
    $,DELIMITER+1))
330 LNUM=FIRST:GOSUB 540:FIRS
    T=P:LNUM=LAST:GOSUB 540:I
    F P=0 THEN P=MAX-1
340 FOR X=FIRST TO P:N$=MID$(
    STR$(LNUM(X)),2)+" "
350 IF CKFLAG=0 THEN A$="":GO
    TO 370
360 CKSUM=0:A$=N$+L$(X):FOR I
    =1 TO LEN(A$):CKSUM=(CKSU
    M+ASC(MID$(A$,I))*I) AND
    255:NEXT:A$=CHR$(65+CKSUM
    /16)+CHR$(65+(CKSUM AND 1
    5))+" "
370 PRINT #1,A$+N$+L$(X)
380 IF INKEY$<>"" THEN X=P
390 NEXT :CLOSE #1:CKFLAG=0
400 GOTO 130
410 IF COMMAND$="LLIST" THEN
    OPEN "lpt1:" FOR OUTPUT A
    S #1:GOTO 300
420 IF COMMAND$="CHECK" THEN
    CKFLAG=1:GOTO 290
430 IF COMMAND$<>"SAVE" THEN
    450
440 GOSUB 600:OPEN ARG$ FOR O
    UTPUT AS #1:ARG$="":GOTO
    300
450 IF COMMAND$<>"LOAD" THEN
    490
460 GOSUB 600:OPEN ARG$ FOR I
    NPUT AS #1:MAX=0:P=0
470 WHILE NOT EOF(1):LINE INP
    UT #1,L$:LNUM(P)=VAL(L$):
    L$(P)=MID$(L$,LEN(STR$(VA
    L(L$)))+1):P=P+1:WEND
480 MAX=P:CLOSE #1:GOTO 130
490 IF COMMAND$="NEW" THEN IN
    PUT "Erase program - Are
    you sure";L$:IF LEFT$(L$,
    1)="y" OR LEFT$(L$,1)="Y"
     THEN MAX=0:GOTO 130:ELSE
     130
500 IF COMMAND$="BASIC" THEN
    COLOR 7,0,0:ON ERROR GOTO
     0:CLS:END
510 IF COMMAND$<>"FILES" THEN
    520
515 IF ARG$="" THEN ARG$="A:"
     ELSE SEL=1:GOSUB 600
517 FILES ARG$:GOTO 130
520 PRINT"Syntax error":GOTO
    130
540 P=0:WHILE LNUM>LNUM(P) AN
    D P<MAX:P=P+1:WEND:RETURN
560 MAX=MAX-1:FOR X=P TO MAX:
    LNUM(X)=LNUM(X+1):L$(X)=L
    $(X+1):NEXT:RETURN
580 MAX=MAX+1:FOR X=MAX TO P+
    1 STEP -1:LNUM(X)=LNUM(X-
    1):L$(X)=L$(X-1):NEXT:L$(
    P)=TEXT$:LNUM(P)=LNUM:RET
    URN
600 IF LEFT$(ARG$,1)<>CHR$(34
    ) THEN 520 ELSE ARG$=MID$
    (ARG$,2)
610 IF RIGHT$(ARG$,1)=CHR$(34
    ) THEN ARG$=LEFT$(ARG$,LE
    N(ARG$)-1)
620 IF SEL=0 AND INSTR(ARG$,"
    .")=0 THEN ARG$=ARG$+".BA
    S"
630 SEL=0:RETURN
640 CLOSE #1:CKFLAG=0:PRINT"S
    topped.":RETURN 150
650 PRINT "Error #";ERR:RESUM
    E 150
```

## Program 4: Apple Proofreader

*By Tim Victor, Editorial Programmer*

```
10 C = 0: FOR I = 768 TO 768 +
   68: READ A:C = C + A: POKE I
   ,A: NEXT
20 IF C < > 7258 THEN PRINT "ER
   ROR IN PROOFREADER DATA STAT
   EMENTS": END
30 IF PEEK (190 * 256) < > 76 T
   HEN POKE 56,0: POKE 57,3: CA
   LL 1002: GOTO 50
40 PRINT CHR$ (4);"IN#A$300"
50 POKE 34,0: HOME : POKE 34,1:
    VTAB 2: PRINT "PROOFREADER
   INSTALLED"
60 NEW
100 DATA 216,32,27,253,201,141
110 DATA 208,60,138,72,169,0
120 DATA 72,189,255,1,201,160
130 DATA 240,8,104,10,125,255
140 DATA 1,105,0,72,202,208
150 DATA 238,104,170,41,15,9
160 DATA 48,201,58,144,2,233
170 DATA 57,141,1,4,138,74
180 DATA 74,74,74,41,15,9
190 DATA 48,201,58,144,2,233
200 DATA 57,141,0,4,104,170
210 DATA 169,141,96
```

# MLX Machine Language Entry Program For Commodore 64 and Apple

*"MLX" allows almost failsafe entry of machine language (ML) programs published in* COMPUTE!. *The Apple version runs on all II-series computers with either DOS 3.3 or ProDOS. The current Commodore 64 version was introduced in the December 1985 issue; no version of 64 MLX published before that date can be used to enter the MLX-format listings published since then.*

Type in and save some copies of the version of MLX for your computer (you'll need it for entering future ML programs in COMPUTE!). For Apple MLX, it doesn't matter whether you save the program on a disk formatted for DOS 3.3 or ProDOS. Programs entered with Apple MLX, however, must be saved to a disk formatted with the same operating system as MLX itself. If you have an Apple IIe or IIc, make sure the CAPS LOCK key is down.

When you're ready to enter an ML program, load and run MLX. It asks you for a starting address and ending address. These addresses appear in the article accompanying the MLX-format program listing you're typing. After you enter the addresses, 64 MLX offers you the option of clearing the workspace. Choose this option only if you're starting to enter a new listing.

A functions menu appears next. The first option is Enter Data. If you're just starting to type in a program, pick this. Begin by typing the first number in the first line of the program listing. If you've already typed in part of a program, type the line number where you left off typing at the end of the previous session. In any case, make sure the address you enter corresponds to the address of a line in the listing you are entering. Otherwise, you'll be unable to enter the data correctly. In 64 MLX, if you select Enter Data by mistake, you can return to the menu by pressing RETURN alone when asked for the address. (You can get back to the menu from most options by pressing RETURN with no other input.)

Once in Enter mode, MLX prints the address for each program line. You then type in all numbers on that line, beginning with the first two-digit number after the colon (:). Each line represents eight data bytes and a checksum. Although an MLX-format listing appears similar to the "hex dump" ML listings you may have seen, the extra checksum number on the end allows MLX to check your typing.

Only the numerals 0–9 and the letters A–F can be typed. If you press any other key (with some exceptions noted below), nothing happens. When you enter a line correctly, MLX adds the data to the workspace area and prompts for the next line (the 64 version also beeps). But if MLX detects a typing error, it notifies you. 64 MLX buzzes and displays an error message, then redisplays the line for editing. Apple MLX beeps, erases the incorrect line, and prompts you to reenter it.

64 MLX formats your input for you, so you may have to unlearn some habits. *Do not* type spaces between the columns; 64 MLX automatically inserts them. *Do not* press RETURN after typing the last number in a line; 64 MLX automatically enters and checks the line after you type the last digit.

Apple MLX is a little different. You can put extra spaces between numbers or leave out the spaces entirely, compressing a line into 18 keypresses. But be careful not to put a space between two digits in the middle of a number. Apple MLX would read two single-digit numbers instead of one two-digit number. You must press RETURN to enter the line.

In 64 MLX, to correct typing mistakes before finishing a line, press INST/DEL to delete the character to the left of the cursor. (The cursor-left key also deletes.) If you mess up a whole line, press CLR/HOME to start the line over. The RETURN key is also active, but only before any data is typed on a line. Pressing RETURN at this point returns you to the command menu. After you type a character of data, 64 MLX disables RETURN until the cursor returns to the start of a line. You can press CLR/HOME to quickly get to a line number prompt.

When 64 MLX detects an error, more editing features become available. Compare the erroneous line on the screen with the one printed in the listing, then move the cursor to the mistake and type the correct key. The cursor-left and -right keys provide the normal cursor controls. (INST/DEL now works as an alternative cursor-left key.) You cannot move left beyond the first character in the line. If you try to move beyond the rightmost character, the line is reentered. During editing, RETURN is active; pressing it tells 64 MLX to recheck the line. You can press CLR/HOME to clear the entire line if you want to start over, or if you want to get to a line number prompt to use RETURN to get back to the menu.

Apple MLX also has editing features. The left- and right-arrow keys let you back up and go forward on the line you're entering so you can retype data. Pressing the CTRL and D keys simultaneously removes the character under the cursor, shortening the line by one character. Pressing CTRL-I inserts a space under the cursor and shifts the rest of the line to the right, making the line one character longer. If the cursor is at the right end of the line, neither CTRL-D nor CTRL-I has any effect. To leave Enter mode, press RETURN when MLX prompts you for a new line.

After you've entered the last number on the last line of the listing, Apple MLX returns to the menu. Immediately choose option S to save your data. 64 MLX automatically moves to the Save option after you type the last number.

Another menu choice, Display Data, shows the contents of memory in the same format as the listing (including the checksum). When you press D, MLX asks you for a starting address. Be sure the address you give matches a line number in the listing. Otherwise, the checksum display is meaningless. MLX displays lines until it reaches the end of the program, then redisplays the menu. With Apple MLX, you can stop the display and return to the menu by pressing any key. 64 MLX lets you stop the display and get back to the menu by pressing RETURN, or pause the display by pressing the space bar (press space again to unpause).

Two more menu selections let you save and load partially typed programs: Save File and Load File in Apple MLX, and Save Data and Load Data in 64 MLX. When you press S or L, MLX asks you for the filename. 64 MLX follows this by asking you to press either D or T for disk or tape. 64 MLX starts and stops the disk drive several times during a load or save; this is normal. Also, 64 MLX automatically adds the drive prefix 0: to the filename, so *do not* include this when entering the filename.

Remember that MLX saves the entire workspace area from the starting address to the ending address, so the save or load may take longer than you might expect if you've entered only a small part of a long listing. When saving a partial listing, make sure to note the address where you stopped typing so you'll know where to resume when you reload.

MLX reports any errors detected during the save or load. 64 MLX displays standard error messages and has

three special load error messages: INCORRECT STARTING ADDRESS, which means the file you're loading does not have the starting address you specified when you ran MLX; LOAD ENDED AT *address*, which means the file you're loading ends before the ending address you specified when you started MLX; and TRUNCATED AT ENDING ADDRESS, which means the file you're loading extends beyond the ending address you specified when you started MLX.

Apple MLX simply displays the message DISK ERROR if it detects a problem during a Save or Load. If you're unsure what caused the error, check the drive. Make sure there's a disk formatted by the same operating system you're using for MLX (ProDOS or DOS 3.3). You'll also see an error message if the disk is full. Either save the file on another disk or quit MLX by pressing the Q key, delete an old file or two, then rerun MLX. Your typing should still be safe in memory. If the error message appears during a Load, make sure the filename exists on disk. An error message when the program isn't trying to access the drive means you've made a typing error in the MLX program itself.

The Quit option stops MLX and enters BASIC. (Of course, RUN/STOP–RESTORE for the 64 or CTRL-RESET for the Apple also quits.) 64 MLX asks for verification; press Y to exit to BASIC, or any other key to return to the menu. After quitting, you can type RUN and reenter MLX without losing your data, as long as you don't use the clear workspace option in 64 MLX.

The instructions for loading and using the finished listing vary from program to program. Some Commodore 64 ML programs are designed to be loaded and run like BASIC programs. Others must be reloaded to specific addresses, then started with a SYS. Always refer to the article which accompanies the ML listing for this information. For the Apple, you need to either BRUN the program, or BLOAD and start the program with a CALL. Again, refer to the article accompanying the program.

For instructions on entering the following listings, please refer to "COMPUTE!'s Guide to Typing In Programs" published in this issue of COMPUTE!.

## Program 1: MLX For Commodore 64

*Version by Ottis Cowper, Technical Editor*

```
100 POKE 56,50:CLR:DIM IN$,I,J
    ,A,B,A$,B$,A(7),N$ :rem 34
110 C4=48:C6=16:C7=7:Z2=2:Z4=2
    54:Z5=255:Z6=256:Z7=127
                       :rem 238
120 FA=PEEK(45)+Z6*PEEK(46):BS
    =PEEK(55)+Z6*PEEK(56):H$="
    0123456789ABCDEF" :rem 118
130 R$=CHR$(13):L$="{LEFT}":S$
    =" ":D$=CHR$(20):Z$=CHR$(0
    ):T$="{13 RIGHT}" :rem 173
140 SD=54272:FOR I=SD TO SD+23
    :POKE I,0:NEXT:POKE SD+24,
    15:POKE 788,52     :rem 194
150 PRINT"{CLR}"CHR$(142)CHR$(
    8):POKE 53280,15:POKE 5328
    1,15               :rem 104
160 PRINT T$" {RED}{RVS}
    {2 SPACES}[8 @3{2 SPACES}"
    SPC(28)"{2 SPACES}{OFF}
    {BLU} MLX II {RED}{RVS}
    {2 SPACES}"SPC(28)"
    {12 SPACES}{BLU}" :rem 121
170 PRINT"{3 DOWN}{3 SPACES}CO
    MPUTE!'S MACHINE LANGUAGE
    {SPACE}EDITOR{3 DOWN}"
                       :rem 135
180 PRINT"{BLK}STARTING ADDRES
    S[43";:GOSUB300:SA=AD:GOSU
    B1040:IF F THEN180:rem 113
190 PRINT"{BLK}{2 SPACES}ENDIN
    G ADDRESS[43";:GOSUB300:EA
    =AD:GOSUB1030:IF F THEN190
                       :rem 173
200 INPUT"{3 DOWN}{BLK}CLEAR W
    ORKSPACE [Y/N][43";A$:IF L
    EFT$(A$,1)<>"Y"THEN220
                       :rem 9
210 PRINT"{2 DOWN}{BLU}WORKING
    ...";:FORI=BS TO BS+EA-SA+
    7:POKE I,0:NEXT:PRINT"DONE
    "                  :rem 139
220 PRINTTAB(10)"{2 DOWN}{BLK}
    {RVS} MLX COMMAND MENU
    {DOWN}[43":PRINT T$"{RVS}E
    {OFF}NTER DATA"    :rem 62
230 PRINT T$"{RVS}D{OFF}ISPLAY
     DATA":PRINT T$"{RVS}L
    {OFF}OAD DATA"     :rem 19
240 PRINT T$"{RVS}S{OFF}AVE FI
    LE":PRINT T$"{RVS}Q{OFF}UI
    T{2 DOWN}{BLK}"    :rem 238
250 GET A$:IF A$=N$ THEN250
                       :rem 127
260 A=0:FOR I=1 TO 5:IF A$=MID
    $("EDLSQ",I,1)THEN A=I:I=5
                       :rem 42
270 NEXT:ON A GOTO420,610,690,
    700,280:GOSUB1060:GOTO250
                       :rem 97
280 PRINT"{RVS} QUIT ":INPUT"
    {DOWN}[43ARE YOU SURE [Y/N
    ]";A$:IF LEFT$(A$,1)<>"Y"T
    HEN220             :rem 189
290 POKE SD+24,0:END   :rem 95
300 IN$=N$:AD=0:INPUTIN$:IFLEN
    (IN$)<>4THENRETURN :rem 31
310 B$=IN$:GOSUB320:AD=A:B$=MI
    D$(IN$,3):GOSUB320:AD=AD*2
    56+A:RETURN        :rem 225
320 A=0:FOR J=1 TO 2:A$=MID$(B
    $,J,1):B=ASC(A$)-C4+(A$>"@
    ")*C7:A=A*C6+B     :rem 143
330 IF B<0 OR B>15 THEN AD=0:A
    =-1:J=2            :rem 132
340 NEXT:RETURN        :rem 240
350 B=INT(A/C6):PRINT MID$(H$,
    B+1,1);:B=A-B*C6:PRINT MID
    $(H$,B+1,1);:RETURN:rem 42
360 A=INT(AD/Z6):GOSUB350:A=AD
    -A*Z6:GOSUB350:PRINT":";
                       :rem 32
370 CK=INT(AD/Z6):CK=AD-Z4*CK+
    Z5*(CK>Z7):GOTO390:rem 131
380 CK=CK*Z2+Z5*(CK>Z7)+A
                       :rem 168
390 CK=CK+Z5*(CK>Z5):RETURN
                       :rem 159
400 PRINT"{DOWN}STARTING AT[43
    ";:GOSUB300:IF IN$<>N$ THE
    N GOSUB1030:IF F THEN400
                       :rem 75
410 RETURN             :rem 117
420 PRINT"{RVS} ENTER DATA ":G
    OSUB400:IF IN$=N$ THEN220
                       :rem 85
430 OPEN3,3:PRINT      :rem 34
440 POKE198,0:GOSUB360:IF F TH
    EN PRINT IN$:PRINT"{UP}
    {5 RIGHT}";        :rem 6
450 FOR I=0 TO 24 STEP 3:B$=S$
    :FOR J=1 TO 2:IF F THEN B$
    =MID$(IN$,I+J,1)   :rem 226
460 PRINT"{RVS}"B$L$;:IF I<24T
    HEN PRINT"{OFF}";  :rem 15
470 GET A$:IF A$=N$ THEN470
                       :rem 135
480 IF(A$>"/"ANDA$<":")OR(A$>"
    @"ANDA$<"G")THEN540
                       :rem 100
490 IF A$=R$ AND((I=0)AND(J=1)
    OR F)THEN PRINT B$;:J=2:NE
    XT:I=24:GOTO550    :rem 46
500 IF A$="{HOME}" THEN PRINT
    {SPACE}B$:J=2:NEXT:I=24:NE
    XT:F=0:GOTO440     :rem 66
510 IF(A$="{RIGHT}")ANDF THENP
    RINT B$L$;:GOTO540:rem 107
520 IF A$<>L$ AND A$<>D$ OR((I
    =0)AND(J=1))THEN GOSUB1060
    :GOTO470           :rem 232
530 A$=L$+S$+L$:PRINT B$L$;:J=
    2-J:IF J THEN PRINT L$;:I=
    I-3                :rem 12
540 PRINT A$;:NEXT J:PRINT S$;
                       :rem 2
550 NEXT I:PRINT:PRINT"{UP}
    {5 RIGHT}";:INPUT#3,IN$:IF
     IN$=N$ THEN CLOSE3:GOTO22
    0                  :rem 106
560 FOR I=1 TO 25 STEP3:B$=MID
    $(IN$,I):GOSUB320:IF I<25
    {SPACE}THEN GOSUB380:A(I/3
    )=A                :rem 81
570 NEXT:IF A<>CK THEN GOSUB10
    60:PRINT"{BLK}{RVS} ERROR:
     REENTER LINE [43":F=1:GOT
    O440               :rem 161
580 GOSUB1080:B=BS+AD-SA:FOR I
    =0 TO 7:POKE B+I,A(I):NEXT
                       :rem 245
590 AD=AD+8:IF AD>EA THEN CLOS
    E3:PRINT"{DOWN}{BLU}** END
     OF ENTRY **{BLK}{2 DOWN}"
    :GOTO700           :rem 207
600 F=0:GOTO440        :rem 84
610 PRINT"{CLR}{DOWN}{RVS} DIS
    PLAY DATA ":GOSUB400:IF IN
    $=N$ THEN220       :rem 146
620 PRINT"{DOWN}{BLU}PRESS:
    {RVS}SPACE{OFF} TO PAUSE,
    {SPACE}{RVS}RETURN{OFF} TO
     BREAK[43{DOWN}"   :rem 241
630 GOSUB360:B=BS+AD-SA:FORI=B
    TO B+7:A=PEEK(I):GOSUB350:
    GOSUB380:PRINT S$; :rem 56
640 NEXT:PRINT"{RVS}";:A=CK:GO
    SUB350:PRINT       :rem 144
650 F=1:AD=AD+8:IF AD>EA THENP
    RINT"{DOWN}{BLU}** END OF
    {SPACE}DATA **":GOTO220
                       :rem 170
660 GET A$:IF A$=R$ THEN GOSUB
    1080:GOTO220       :rem 65
670 IF A$=S$ THEN F=F+1:GOSUB1
    080                :rem 28
680 ONFGOTO630,660,630:rem 224
```

```
690 PRINT"{DOWN}{RVS} LOAD DAT
    A ":OP=1:GOTO710    :rem 31
700 PRINT"{DOWN}{RVS} SAVE FIL
    E ":OP=0            :rem 32
710 IN$=N$:INPUT"{DOWN}FILENAM
    E[43]";IN$:IF IN$=N$ THEN22
    0                   :rem 229
720 F=0:PRINT"{DOWN}{BLK}{RVS}
    T{OFF}APE OR {RVS}D{OFF}IS
    K: [43]";           :rem 66
730 GET A$:IF A$="T"THEN PRINT
    "T{DOWN}":GOTO880   :rem 90
740 IF A$<>"D"THEN730   :rem 90
750 PRINT"D{DOWN}":OPEN15,8,15
    ,"I0:":B=EA-SA:IN$="0:"+IN
    $:IF OP THEN810     :rem 163
760 OPEN 1,8,8,IN$+",P,W":GOSU
    B860:IF A THEN220   :rem 66
770 AH=INT(SA/256):AL=SA-(AH*2
    56):PRINT#1,CHR$(AL);CHR$(
    AH);                :rem 221
780 FOR I=0 TO B:PRINT#1,CHR$(
    PEEK(BS+I));:IF ST THEN800
                        :rem 171
790 NEXT:CLOSE1:CLOSE15:GOTO94
    0                   :rem 230
800 GOSUB1060:PRINT"{DOWN}
    {BLK}ERROR DURING SAVE:[43]
    ":GOSUB860:GOTO220  :rem 61
810 OPEN 1,8,8,IN$+",P,R":GOSU
    B860:IF A THEN220   :rem 57
820 GET#1,A$,B$:AD=ASC(A$+Z$)+
    256*ASC(B$+Z$):IF AD<>SA T
    HEN F=1:GOTO850     :rem 155
830 FOR I=0 TO B:GET#1,A$:POKE
     BS+I,ASC(A$+Z$):IF ST AND
    (I<>B)THEN F=2:AD=I:I=B
                        :rem 180
840 NEXT:IF ST<>64 THEN F=3
                        :rem 20
850 CLOSE1:CLOSE15:ON ABS(F>0)
    +1 GOTO960,970      :rem 12
860 INPUT#15,A,A$:IF A THEN CL
    OSE1:CLOSE15:GOSUB1060:PRI
    NT"{RVS}ERROR: "A$:rem 114
870 RETURN              :rem 127
880 POKE183,PEEK(FA+2):POKE187
    ,PEEK(FA+3):POKE188,PEEK(F
    A+4):IFOP=0THEN920:rem 178
890 SYS 63466:IF(PEEK(783)AND1
    )THEN GOSUB1060:PRINT"
    {DOWN}{RVS} FILE NOT FOUND
     ":GOTO690          :rem 34
900 AD=PEEK(829)+256*PEEK(830)
    :IF AD<>SA THEN F=1:GOTO97
    0                   :rem 201
910 A=PEEK(831)+256*PEEK(832)-
    1:F=F-2*(A<EA)-3*(A>EA):AD
    =A-AD:GOTO930       :rem 75
920 A=SA:B=EA+1:GOSUB1010:POKE
    780,3:SYS 63338     :rem 107
930 A=BS:B=BS+(EA-SA)+1:GOSUB1
    010:ON OP GOTO950:SYS 6359
    1                   :rem 38
940 GOSUB1080:PRINT"{BLU}** SA
    VE COMPLETED **":GOTO220
                        :rem 139
950 POKE147,0:SYS 63562:IF ST<
    >64 THEN970         :rem 39
960 GOSUB1080:PRINT"{BLU}** LO
    AD COMPLETED **":GOTO220
                        :rem 126
970 GOSUB1060:PRINT"{BLK}{RVS}
    ERROR DURING LOAD:{DOWN}
    [43]":ON F GOSUB980,990,100
    0:GOTO220           :rem 233
980 PRINT"INCORRECT STARTING A
    DDRESS (";:GOSUB360:PRINT"
    )":RETURN           :rem 145
990 PRINT"LOAD ENDED AT ";:AD=
    SA+AD:GOSUB360:PRINT D$:RE
    TURN                :rem 159
1000 PRINT"TRUNCATED AT ENDING
     ADDRESS":RETURN    :rem 166
1010 AH=INT(A/256):AL=A-(AH*25
     6):POKE193,AL:POKE194,AH
                        :rem 95
1020 AH=INT(B/256):AL=B-(AH*25
     6):POKE174,AL:POKE175,AH:
     RETURN             :rem 122
1030 IF AD<SA OR AD>EA THEN105
     0                  :rem 135
1040 IF(AD>511 AND AD<40960)OR
     (AD>49151 AND AD<53248)TH
     EN GOSUB1080:F=0:RETURN
                        :rem 104
1050 GOSUB1060:PRINT"{RVS} INV
     ALID ADDRESS {DOWN}{BLK}"
     :F=1:RETURN        :rem 224
1060 POKE SD+5,31:POKE SD+6,20
     8:POKE SD,240:POKE SD+1,4
     :POKE SD+4,33      :rem 19
1070 FOR S=1 TO 100:NEXT:GOTO1
     090                :rem 90
1080 POKE SD+5,8:POKE SD+6,240
     :POKE SD,0:POKE SD+1,90:P
     OKE SD+4,17        :rem 182
1090 FOR S=1 TO 100:NEXT:POKE
     {SPACE}SD+4,0:POKE SD,0:P
     OKE SD+1,0:RETURN  :rem 8
```

## Program 2: MLX For Apple

*Version by Tim Victor, Editorial Programmer*

```
100 N = 9: HOME : NORMAL : PRIN
    T "APPLE MLX": POKE 34,2: O
    NERR  GOTO 610
110 VTAB 1: HTAB 20: PRINT "STA
    RT ADDRESS";: GOSUB 530: IF
     A = 0 THEN  PRINT  CHR$ (7
    ): GOTO 110
120 S = A
130 VTAB 2: HTAB 20: PRINT "END
     ADDRESS  ";: GOSUB 530: IF
     S  = A OR A = 0 THEN  PR
    INT  CHR$ (7): GOTO 130
140 E = A
150 PRINT : PRINT "CHOOSE: (E)NT
    ER DATA";: HTAB 22: PRINT "
    (D)ISPLAY DATA": HTAB 8: PR
    INT "(L)OAD FILE  (S)AVE FI
    LE  (Q)UIT": PRINT
160 GET A$: FOR I = 1 TO 5: IF
    A$ < > MID$ ("EDLSQ",I,1) T
    HEN  NEXT : GOTO 160
170 ON I GOTO 270,220,180,200:
    POKE 34,0: END
180 INPUT "FILENAME: ";A$: IF A
    $ <  > "" THEN  PRINT  CHR$
     (4);"BLOAD";A$;",A";S
190 GOTO 150
200 INPUT "FILENAME: ";A$: IF A
    $ <  > "" THEN  PRINT  CHR$
     (4);"BSAVE";A$;",A";S;",L"
    ;E - S
210 GOTO 150
220 GOSUB 590: IF B = 0 THEN 15
    0
230 FOR B = B TO E STEP 8:L = 4
    :A = B: GOSUB 580: PRINT A$
    ;": ";:L = 2
240 FOR F = 0 TO 7:V(F + 1) = P
    EEK (B + F): NEXT : GOSUB 5
    60:V(9) = C
250 FOR F = 1 TO N:A = V(F): GO
    SUB 580: PRINT A$" ";: NEXT
     : PRINT : IF  PEEK (49152)
     < 128 THEN  NEXT
260 POKE 49168,0: GOTO 150
270 GOSUB 590: IF B = 0 THEN 15
    0
280 FOR B = B TO E STEP 8
290 HTAB 1:A = B:L = 4: GOSUB 5
    80: PRINT A$;": ";: CALL 64
    668:A$ = "":P = 0: GOSUB 33
    0: IF L = 0 THEN 150
300 GOSUB 470: IF F < > N THEN
    PRINT CHR$ (7);: GOTO 290
310 IF N = 9 THEN GOSUB 560: IF
     C < > V(9) THEN PRINT CHR$
     (7);: GOTO 290
320 FOR F = 1 TO 8: POKE B + F
    - 1,V(F): NEXT : PRINT : NE
    XT : GOTO 150
330 IF LEN (A$) = 33 THEN A$ =
    O$:P = O: PRINT CHR$ (7);
340 L = LEN (A$):O$ = A$:O = P:
    L$ = "": IF P > 0 THEN L$ =
     LEFT$ (A$,P)
350 R$ = "": IF P < L - 1 THEN
    R$ =  RIGHT$ (A$,L - P - 1)
360 HTAB 7: PRINT L$;: FLASH :
    IF P < L THEN PRINT MID$ (A
    $,P + 1,1);: NORMAL : PRINT
    R$;
370 PRINT " ";: NORMAL
380 K = PEEK (49152): IF K < 12
    8 THEN 380
390 POKE 49168,0:K = K - 128
400 IF K = 13 THEN HTAB 7: PRIN
    T A$;" ";: RETURN
410 IF K = 32 OR K > 47 AND K <
     58 OR K > 64 AND K < 71 TH
    EN A$ = L$ + CHR$ (K) + R$:
    P = P + 1
420 IF K = 4 THEN A$ = L$ + R$
430 IF K = 9 THEN A$ = L$ + " "
     +  MID$ (A$,P + 1,1) + R$
440 IF K = 8 THEN P = P - (P >
    0)
450 IF K = 21 THEN P = P + (P <
    L)
460 GOTO 330
470 F = 1:D = 0: FOR P = 1 TO L
    EN (A$):C$ = MID$ (A$,P,1):
     IF F > N AND C$ < > " " TH
    EN  RETURN
480 IF C$ < > " " THEN  GOSUB 5
    20:V(F) = J + 16 * (D = 1)
    * V(F):D = D + 1
490 IF D > 0 AND C$ = " " OR D
    = 2 THEN D = 0:F = F + 1
500 NEXT : IF D = 0 THEN F = F
    - 1
510 RETURN
520 J = ASC (C$):J = J - 48 - 7
    * (J > 64): RETURN
530 A = 0: INPUT A$:A$ = LEFT$
    (A$,4): IF LEN (A$) = 0 THE
    N RETURN
540 FOR P = 1 TO LEN (A$):C$ =
    MID$ (A$,P,1): IF C$ < "0"
    OR C$ > "9" AND C$ < "A" OR
     C$ > "Z" THEN A = 0: RETUR
    N
550 GOSUB 520:A = A * 16 + J: N
    EXT : RETURN
560 C = INT (B / 256):C = B - 2
    54 * C - 255 * (C > 127):C
    = C - 255 * (C > 255)
570 FOR F = 1 TO 8:C = C * 2 -
    255 * (C > 127) + V(F):C =
    C - 255 * (C > 255): NEXT :
     RETURN
580 I = FRE (0):A$ = "": FOR I
    = 1 TO L:T =  INT (A / 16):
    A$ =  MID$ ("0123456789ABCD
    EF",A - 16 * T + 1,1) + A$:
    A = T: NEXT : RETURN
590 PRINT "FROM ADDRESS ";: GOS
    UB 530: IF S > A OR E < A O
    R A = 0 THEN B = 0: RETURN
600 B = S + 8 * INT ((A - S) /
    8): RETURN
610 PRINT "DISK ERROR": GOTO 15
    0                                   ©
```

# Classified

## SOFTWARE

THE INVESTMENT MANAGER a program for the 64 guaranteed to out-perform any other method. Comes with two programs to help your investment plan. All three $19.95 or write for free details to: Author's Club Software, 6027 S. High, Suite 410, Oklahoma City, OK 73149

TI-99/4A QUALITY SOFTWARE for Business, Home and Entertainment ** BONUS Software Offer! ** Send for FREE Catalog to MICRO-BIZ HAWAII, Box 1108 Pearl City, HI 96782

TI-99/4A Software/Hardware bargains. Hard-to-find items. Huge selection. Fast service. Free catalog. D.E.C., Box 690, Hicksville, NY 11801

BIBLE QUIZ GAMES and other Bible software for the C64. Fun and learn for all ages. BIBLE-MATCH-WITS I (easy), II (hard). STAIRWAY TO HEAVEN 1, 2, 3. Colorful, animation, graphics, and sound. Each $29.95. Order or send SASE for brochure to COMPEDS, P.O. Box 147, Narrows, VA 24124

STAR TRADER a program for 2 to 5 players. Show off your 64 when guests come over. Gomuko compiled basic program 15 by 15 board, Unmove, load/save game user changable logic $14.95 each. Author's Club Software, 6027 S. High, Suite 410, Oklahoma City, OK 73149

**/SPEEDPAK/-C64 Speedscript 3.0 + enhancer. Adds alt. screens, macros, encryption, help screen, chr. set, mail merge & more! Disk. $15 to: /SPEEDPAK/ P.O.B. 22022, Dept. C1, Greensboro, NC 27420**

COMPUQUEST ADVENTURES, Direct for less! 48K Atari (D), Apple (D), C64 (D or T), 48K Spectrum (T). New Special: "Transported" $12.95 + $3 s/h (WI res. 5% tax) to: COMPUQUEST, Box A492, St. Croix Falls, WI 54024. SASE for info.

**MORSE CODE COURSE FOR COMMODORE 64**
for the beginner as well as the AMATEUR who wants to improve speed. For info: F. Ardavin, 1254 Sweet Pine Dr., Norcross, Georgia 30093. Or send $19.95 for disk.

MEDICAL COMPUTER PROGRAMS in basic, $20/booklet, Alan Kwasman, MD, 2844 Taurus, Riverside, CA 92503 (714) 785-8957

TI-99/4A - 130 PROGRAMS $3 EACH! Catalog $1, refunded. Nuts & Bolts, diskfull of 100 mergeable utilities $19.95 ppd. Tigercub Tips, diskfull of 50 programs & C $15 ppd. TIGERCUB, 156 Collingwood, Columbus, OH 43213

FANTASTIC DAILY NUMBER FORECASTER! Guaranteed Winners or Money Back! Picks up to 3 STRAIGHT WINNERS most every week, playing 1 to 3 a day! Apple, IBM, C64, Atari, 1 drive. Many reports of hitting for THOU$AND$. Send SASE for info. $99.95 on disk only to: Z-Way, P.O. Box 9017, Canton, OH 44711

APPLE, ATARI AND COMMODORE OWNERS!!! DISCOUNT PRICES ON NAME BRAND SOFTWARE. SEND SASE FOR PRICE LISTING. SOFTWARE BROTHERS, POB 07095, MILW. WI 53207

**TI-99/4A NEW STATES AND CAPITALS GAME**
Hi-Res map of USA. Send $12 for cass. Or $1 for more info. to: TRINITY SYSTEMS 1022 Grandview, Pittsburgh, PA 15237

PROGRAMS FOR THE TANDY 1000
Send $1 for list of educat. & entertain. progs. Refundable with first purchase. SODA POP SW, POB 653, Kenosha, WI 53141

MAILBOX - Fun, easy, efficient electronic mail for single station, multiuser C-64; passwords, printable output & more; also a great game for kids; disk only $7.95. Pete Kvale, 3156 Choctaw, Memphis, TN 38111

## HARDWARE

IBM PC for Christmas? Build a compatible for a fraction of the cost. Easy to assemble. Write for free datalogue to JV Systems, P.O. Box 9807, Brook Park, OH 44142 or call (216) 842-4313

COMMODORE C128 80-CHARACTER CABLE. No need for RGB1 monitor. 80 char. on reg. monitor. Just $9.95 + $2 s/h to: UNITED RESEARCH, 7723 R'Horse Ln., Boerne, TX 78006

## MISCELLANEOUS

64 AUTHOR'S CLUB - We get you published. Send for free demo-disk. Send $5.00 shipping and handling to: Author's Club, 6027 S. High, Suite 410, Oklahoma City, OK 73149

**HELP IS ON THE WAY!**
Just call 1-800-334-0868 to get your free copy of the latest COMPUTE! Books Catalog! If you need help in getting information on all of the latest COMPUTE! book titles available plus all COMPUTE! backlist titles, call us today!

*** MR. SOFTWARE CO. ALL POPULAR TITLES ***
* Printers, Monitors, Drives, VISA, MC * heavy discounts - Send $1.00 for catalog 11-9 Exton Complex, Somers Point, NJ 08244

HACKER'S HELPER holds magazine or book upright and keeps your place. It's like having three extra hands. Send $24.95 to John Keener, RD#3, Butler, PA 16001

C64 USERS - FREE BROCHURE! Game and instructional programs, each include detailed analysis, beg. or int. level. SASE to: C16 H.O.S, 19730 Ave 18, Madera, CA 93637

FREE CATALOG - TI-99, COMMODORE, IBM. SPECIFY. Hardware, Software, Accessories. Competition Computer, 2629 W. National, Milwaukee, WI 53204 (800) 662-9253

**1541 PERMANENT ALIGNMENT $35.**
**C64 repair $55, 1541 repair $85.**
**DI-TECH, 701 East North Ave., #C**
**Lompoc, CA 93436 (805) 736-9727**

Maxell MD1, $1.29-MD2, $1.99. Dysan 104/1D, $1.79-104/2D, $2.39. Shipping $3.75. Also Verbatim, IBM, 3M, BASF. TAPE WORLD, 220 Spring St., Butler, PA 16001, 1-800-245-6000. Visa, MC.

DISK SALE! - SS/DD 35-trk for Apple w/sleeve & label-10/$5.80, bulk-100/$45. Standard SS/DD w/sleeve & label-10/$7.50, bulk-100/$59. DS/DD w/sleeve & label-10/$8.50, bulk-100/$67. 3½" SS for Mac-10/$19.99. PREMIUM QUALITY, LIFETIME WARRANTY! Money-back satisfaction guarantee! Min. order $20. Send check or pay by MC/VISA/AE $3 shipping, + $2 if C.O.D. - UNITECH, 20 Hurley St., Cambridge, MA 02141. (800) 343-0472, in Mass. (617) "UNI-TECH".

### COMPUTE! Classified is a low-cost way to tell over 350,000 microcomputer owners about your product or service.

**Rates:** $25 per line, minimum of four lines. Any or all of the first line set in capital letters at no charge. Add $15 per line for boldface words, or $50 for the entire ad set in boldface (any number of lines.)

**Terms:** Prepayment is required. Check, money order, American Express, Visa, or MasterCard is accepted. Make checks payable to COMPUTE! Publications.

**Form:** Ads are subject to publisher's approval and must be either typed or legibly printed. One line equals 40 letters and spaces between words. Please underline words to be set in boldface.

**General Information:** Advertisers using post office box numbers in their ads must supply permanent address and telephone numbers. Orders will not be acknowledged. Ad will appear in next available issue after receipt.

**Closing:** 10th of the third month preceding cover date (e.g., June issue closes March 10th). Send order and remittance to: Harry Blair, Classified Manager, COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. To place an ad by phone, call Harry Blair at (919) 275-9809.

**Notice:** COMPUTE! Publications cannot be responsible for offers or claims of advertisers, but will attempt to screen out misleading or questionable copy.

## Statement of Ownership, Management, and Circulation as Required by 39 U.S.C. 3685

1A. COMPUTE!
1B. 537250
2. 9-27-85
3. Monthly
3A. Twelve
3B. $24.00
4. 324 West Wendover Ave., Suite 200, Greensboro, NC 27408
5. Same
6. Publisher, James Casella, 825 Seventh Avenue, New York, NY 10019
Editor, Robert C. Lock, 324 W. Wendover Ave., Suite 200, Greensboro, NC 27408
Managing Editor, Kathleen Martinek, 324 W. Wendover Ave., Suite 200, Greensboro, NC 27408
7. American Broadcasting Companies, Inc., 1330 Ave. of the Americas, New York, NY 10019
8. N/A
9. N/A
10. Extent and Nature of Circulation

| | Average no. of copies each issue during preceding 12 months | Actual no. copies of single issue published nearest to filing date |
|---|---|---|
| A. Total no. Copies (Net Press Run) | 552,208 | 535,075 |
| B. Paid Circulation | | |
| 1. Sales through dealers and carriers, street vendors, and counter sales | 138,289 | 105,889 |
| 2. Mail subscriptions | 210,410 | 247,404 |
| C. Total Paid Circulation | 348,699 | 353,293 |
| D. Free Distribution by mail, carrier, or other means, samples, complimentary and other free copies | 440 | 1,049 |
| E. Total Distribution | 349,139 | 354,342 |
| F. Copies not Distributed | | |
| 1. Office use, left over, unaccounted for, spoiled after printing | 18,453 | 7,232 |
| 2. Returns from news agents | 184,616 | 173,501 |
| G. Total | 552,208 | 535,075 |

I certify that the statements made by me above are correct and complete, James Casella, Publisher.

# Advertisers Index

## COMPUTE!'s FREE Reader Information Service

Use these cards to request FREE information about the products advertised in this issue. Clearly print or type your full name and address. Only one card should be used per person. Circle the numbers that correspond to the key number appearing in the advertisers index.

Send in the card and the advertisers will receive your inquiry. Although every effort is made to insure that only advertisers wishing to provide product information have reader service numbers, COMPUTE! cannot be responsible if advertisers do not provide literature to readers.

Please use these cards *only* for subscribing or for requesting product information. Editorial and customer service inquiries should be addressed to: COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Check the expiration date on the card to insure proper handling.

**Use these cards and this address only for COMPUTE!'s Reader Information Service. Do not send with payment in any form.**

# COMPUTE!

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 |
| 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 |
| 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 |
| 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 |
| 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 |
| 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 |
| 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 |
| 220 | 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 |
| 237 | 238 | 239 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 |

Please let us know. Do you own: plan to buy:

- ☐ 270 Apple _____ ☐ 271
- ☐ 272 Atari _____ ☐ 273
- ☐ 274 Commodore _____ ☐ 275
- ☐ 276 IBM _____ ☐ 277
- ☐ 278 TI-99/4A _____ ☐ 279
- ☐ 280 Other _____ (specify model) ☐ 281

Please print or type name and address.
Limit one card per person.

Name _____

Address _____

City _____

State/Province _____ Zip _____

Country _____

Please Include ZIP Code Expiration 2/28/86 CO186

# SUBSCRIBE TO COMPUTE!

For Fastest Service, Call Our **Toll-Free** US Order Line
**800-334-0868**
**In NC call 919-275-9809**

My Computer Is:

01 ☐ Apple 02 ☐ Atari 03 ☐ Commodore 64
04 ☐ VIC-20 05 ☐ IBM 06 ☐ TI-99/4A
99 ☐ Other _____ ☐ Don't yet have one.

☐ $18.00 One Year US Subscription
☐ $36.00 Two Year US Subscription

(Readers outside of the US, please see our foreign readers subscription card or inquire for rates).

Name _____

Address _____

City _____ State _____ Zip _____

☐ Payment Enclosed ☐ Bill me
Charge my: ☐ VISA ☐ MasterCard ☐ American Express
Account No. _____ Expires ____ / ____

Your subscription will begin with the next available issue. Please allow 4-6 weeks for delivery. Foreign and Canadian rates subject to change at any time.
The COMPUTE! subscriber list is made available to carefully screened organizations with a product or service which may be of

Place
Stamp
Here

**COMPUTE! Reader Service**
P.O. Box 2141
Radnor, PA 19089

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**
FIRST CLASS PERMIT NO. 7478 DES MOINES, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

**COMPUTE!**
PO BOX 10954
DES MOINES, IOWA 50347

# Reserve my FREE Hour and Rush my Viewtron® Starter Kit

**To order, mail this card or call anytime:** 1 (800) 543-5500, Dept. 9014.

Name _____
(Please print)

Address _____

City/State/Zip _____

Phone Number (_____) _____

Credit cards only; sorry, no checks.

Charge to: ☐ VISA ☐ MasterCard ☐ American Express

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ ☐☐ ☐☐

Card Number Expiration Date

Signature: _____

Yes, send me one of the Viewtron products checked below.

| Description | Cost | Check Here |
|---|---|---|
| Viewtron Starter Kit for my: | | |
| Commodore 64 or 128 | $ 9.95 | ☐ |
| Apple IIe, II + or IIc | $ 9.95 | ☐ |
| IBM PC or PC Compatible | $ 9.95 | ☐ |
| Other type of computer. I already have communications software which includes VT100 emulation, such as "Crosstalk" or "MacTerminal" | $ 9.95 | ☐ |
| MODEM PACKAGE FOR COMMODORE 64 or 128 | | |
| 300 Baud Volks 6420 by Anchor Automation with FREE Viewtron Starter Kit | $ 49.95 | ☐ |
| MODEM PACKAGES FOR APPLE IIs | | |
| I have an Apple ☐ IIe ☐ II + ☐ IIc | | |
| 300 Baud Signalman Mark X with FREE Viewtron Starter Kit | $ 99.95 | ☐ |
| 300 Baud Signalman Mark X, and Apricorn serial card (for use with Apple II + or IIe) with FREE Viewtron Starter Kit | $159.90 | ☐ |

**Satisfaction Guaranteed** or your money back. All modem offers include cable (where necessary). If for any reason you are not satisfied, send back the merchandise within 30 days to receive a full refund. Sorry, no C.O.D.'s. Allow 2-4 weeks for delivery. Shipping included. Florida, Illinois and New York residents will have the appropriate sales tax added to their order.

Offers expire February 28, 1986.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

# BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 19696 MIAMI, FL

Postage will be paid by addressee

**Viewtron**®

The New, Easier-to-Use On-Line Service
Viewdata Corporation of America, Inc.
P.O. Box 317678
Cincinnati, Ohio 45231

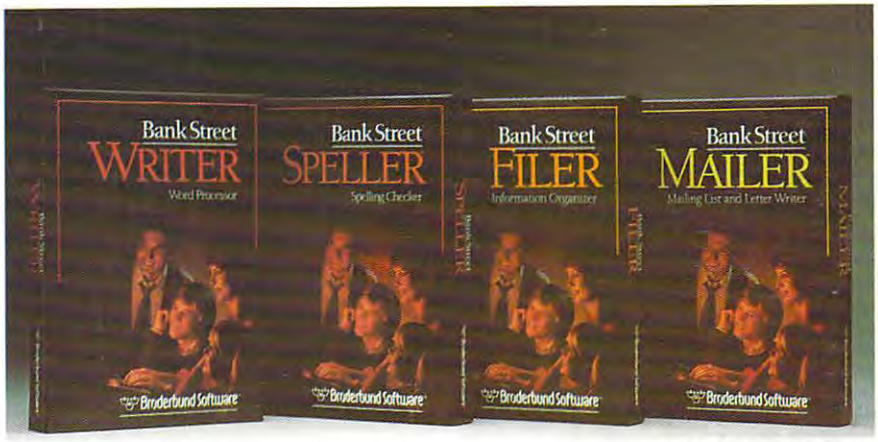
# ONE GOOD THING LEADS *to* ANOTHER.

PRODUCTIVITY SOFTWARE ISN'T VERY PRODUCTIVE IF it's so complicated to learn, so complicated to use, that it never is used.

So when we sat down to design the Bank Street Writer,™ we kept one important objective in mind: to combine all the powerful features that people need in an affordably-priced word processor and make it so easy to use that just about anyone will be productive in moments.

We must have succeeded. The Bank Street Writer is the number one word processing choice of over 300,000 users worldwide.

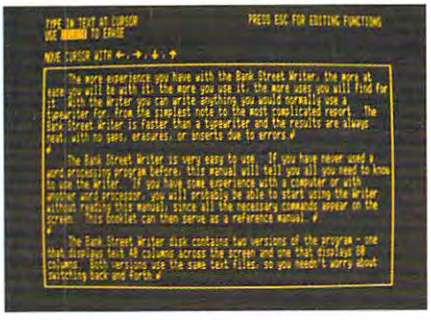
Now, that same philosophy can be found in a complete series of productivity software: the Bank Street Speller,™




BANK STREET PRODUCTS FROM BRØDERBUND MAKE IT EASY TO MAKE YOUR COMPUTER WORK HARD.

Bank Street Filer™ and Bank Street Mailer.™ All the features you'll ever need. None of the complications you don't need. The perfect complements to the Bank Street Writer.

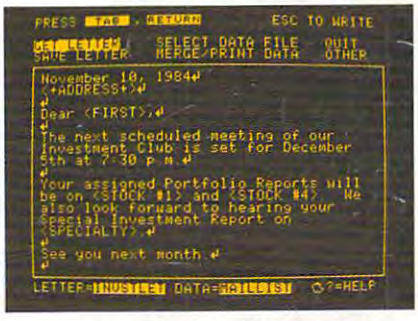
Bank Street products from Brøderbund give you more power for less money with less hassle. And when you think of it, that's the best kind of "productivity."



THE BANK STREET WRITER *lets you write letters, memos, articles or lengthy reports, better and faster. Continually enhanced and updated since its introduction three years ago, the Bank Street Writer is packed with features usually found only in far more expensive programs.*

THE BANK STREET SPELLER *finds and highlights spelling errors and suggests correct spellings; proofreads even the longest documents in seconds. The 31,000 word electronic dictionary can be amended with your own entries, including special terms, trademarks and proper names.*


THE BANK STREET FILER *helps you organize information and print out custom reports in moments. Collect, explore, organize and manipulate data in a variety of natural and flexible ways. For stamp collections or small business record keeping... for home financial and tax data... for bibliographies and reference files... for just about any kind of information you want to store and retrieve, the Bank Street Filer is as simple as using a file cabinet—only much faster, more convenient and more flexible.*

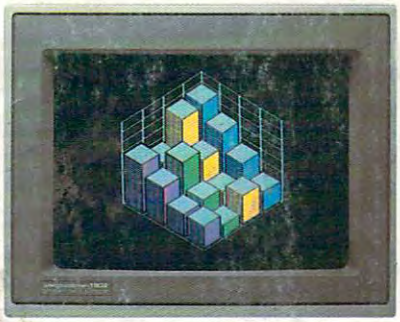



THE BANK STREET MAILER. *Whether you write occasional letters, produce a monthly newsletter or send out mailings to a long list of clients, the Bank Street Mailer does it quickly, easily and efficiently. You can insert names or addresses into a form letter, or send a personalized mailing to customers sorted by zip code, street address or any other aspect of your files. The Mailer can be used by itself (with its own built-in letter-writer) or with letters and lists from the Bank Street Writer and Filer.*

Brøderbund®

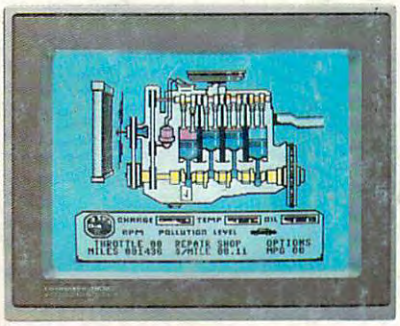
Versions of the BANK STREET WRITER are available for Apple, IBM-PC/PCjr, Commodore 64 and Atari personal computers. The BANK STREET SPELLER, BANK STREET FILER and BANK STREET MAILER are available for Apple and Commodore 64. For more information about Brøderbund products, please write us at 17 Paul Drive, San Rafael, California 94903-2101. Apple, IBM PCjr, Commodore and Atari are registered trademarks of Apple Computer, Inc., International Business Machines Corporation, Commodore Electronics, Ltd. and Atari Corporation respectively. 

# All you need to do this



graph a spreadsheet

write a novel



fix an engine

compose a song

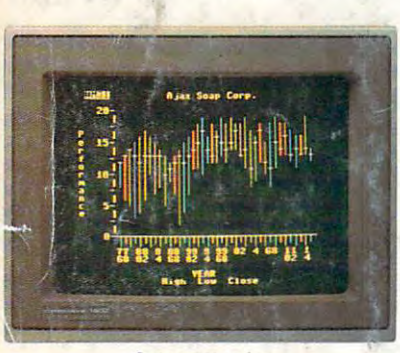
paint a picture

ur banking

learn to fly

organize a data base

tell a story




forecast sales


When it comes to personal computers, you want the smartest you can own. At a price that makes sense.

The new Commodore 128™ system has a powerful 128K memory, expandable to 512K. An 80-column display and 64, 128 and CP/M® modes for easy access to thousands of educational, business and home programs. And a keyboard, with built-in numeric keypad, that operates with little effort.

Discover the personal computer that does more for you. At the price you've been waiting for. From the company that sells more personal computers than IBM® or Apple®.



**COMMODORE 128 PERSONAL COMPUTER**

A Higher Intelligence

www.commodore.ca