NO. 71 PRINTER CONTROL

U.S. Edition: $2.50
International Edition: $3.00

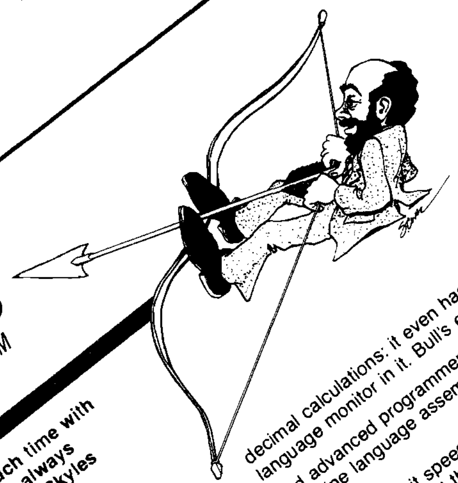
Complete Programs Included

MICRO™

for the Serious Computerist

PRINTER CONTROL

* Double Vision
* Micro Mouse
* Easy DOES-IT
* The Accurate Printer
* HiRes Screen Dump



Shoot the Works With an Arrow™

Have you been spending too much time with your *VIC*? If it's because you are always waiting for your programs to load, Skyles Electric Works can help.

Enter the ARROW

This cartridge can load programs off cassette at about eight times the normal speed. It can record a program once at up to eight times the density on tape. This means that you can get 200 kbytes of programs on a single C-20 tape. In the long run, you will be able to put 10 tapes into one.

The *ARROW* cartridge will pay for itself in the tape and the vast amount of loading time it saves.

. . . and it's easy to use

ARROW . . . . . adds 4 commands: S, L, V, and A that are used to SAVE, LOAD, VERIFY and APPEND at a high speed (3600 baud).

ARROW . . . . . leaves the usual messages that appear on the screen.

ARROW . . . . . adds a message telling you the length of the program that has been loaded (or saved).

ARROW . . . . . keeps all the usual shorthand features of program naming.

ARROW . . . . . starts automatically when you turn on your *VIC-20* or *CBM-64*.

It gives the length of the program that it has loaded or saved. It can handle hexadecimal/decimal calculations: it even has a machine language monitor in it. Bull's eye!

And advanced programmers can get an optional machine language assembler in the same cartridge.

Even though it speeds up program loading it does not effect the tape recorder. It's an extremely reliable device.

But this great device does more than cut down on loading time and tape space. It is a fine programmer's aid, too. With some sharp but simple program text editing commands, the *ARROW* might be the best bargain around in personal software for your *VIC-20*. The *ARROW* is always on target with auto line number and line deleting features.

Only $49.95 for the *VIC-20* or *COMMODORE 64* version. The high-speed cartridge with machine language assembler costs just $89.95. And the easy-to-use *ARROW* comes with the clear Skyles manual.

If you want lightning-fast cassette operations, shoot for the *ARROW*.

AR20 (VIC-20), AR64 (CBM-64) **$49.95***
ARAS20 (VIC-20), ARAS64 (CBM-64) . . **89.95***

If your programs are slow, *BLITZ!*™ them

Bob has discovered a spectacular compiler for the *COMMODORE 64* that he calls BLITZ!

BLITZ! is faster than *PET SPEED*, faster than any other Commodore compiler that has come down the pike. Your *BLITZ!*-compiled program will run from *five to 20 times* faster than it did before you blitzed it.

This disk program translates your slow BASIC programs into a much, much faster code. The *BLITZ!* compiler significantly improves the performance of your BASIC routines. It reads the entire program, reduces that program's size by deciding which operations need only run once, interprets the operations, and then re-writes the program into its own special P-code.

For more advanced programmers, the *BLITZ!* compiler offers another benefit besides speed. It provides an almost infinite number of ways to add protection to your programs, because a compiled program is not readable. So security is inherent in the re-writing!

Using this compiler, you can now create a string of BASIC programs such that one automatically loads the next. You can also pass information from one program to another. You will find the *BLITZ!* compiler a treasured tool in your programming kit.

In disk, for the *COMMODORE 64*, *BLITZ!* Bob can also get you one for the older *PET CBMs*, on a special-order basis.

Don't wait all day for your programs to run. *BLITZ!* them.

BLITZ, COMMODORE 64, disk **$99.00***

*There is an additional $3.50 US and Canada, $10.00 Europe and Asia, shipping charge per order. California residents add sales tax.

Skyles Catalogue Page 2

For CBM-64 and VIC owners only:

This is just 1 of 20 pages of the newest and biggest Skyles catalog, hot off the press.

We know you'll want this page, in its full 7 x 10 splendor, and another 19 pages of peripherals, software and books that will make your CBM-64 or VIC computer even nicer to live with. So, if we missed sending you your very own copy within the last few weeks, call us at **(800) 227-9998** **(unless you live in California, in which case call (415) 965-1735.**

From Skyles Electric Works, the oldest and largest professional specialists in the business.

Skyles Electric Works
231-E S. Whisman Road
Mountain View, California 94041

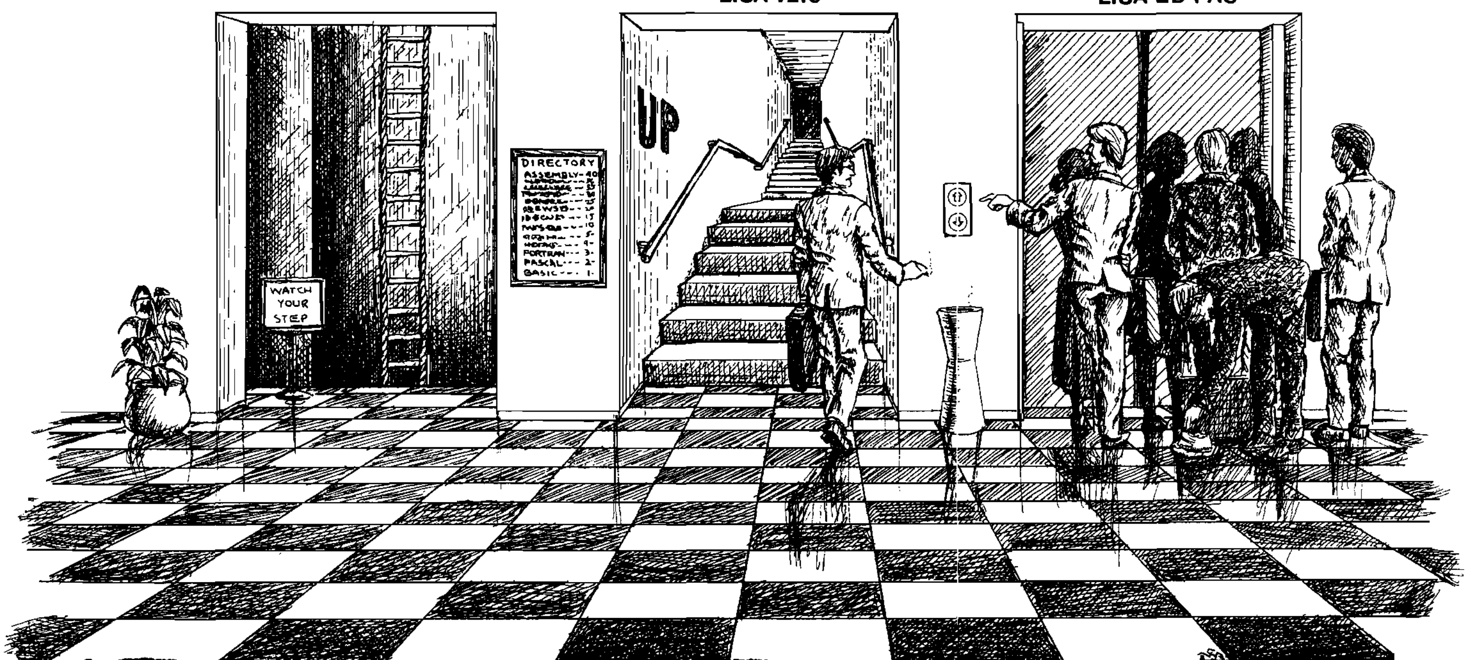
There are three ways to learn 6502 Assembly Language on your Apple Computer:

Hard

Easy

Easiest

OTHER ASSEMBLERS | LISA v2.6 | LISA ED PAC



Introducing the Easiest Way: The LISA Ed Pac™

You can't deny that learning assembly is extremely important for you if you want to make the most of your work. If assembly language wasn't so important, why are almost all of the top selling programs available for the Apple II written in assembly language? But let's face it, learning 6502 assembly language isn't a piece of cake. At least not until now. Because now there's the LISA Education Package™ from Lazerware. It'll have you up to speed with assembly language in a fraction of the time it would otherwise take.

The LISA Ed Pac™ begins with LISA v2.6, the favorite assembler of beginners and professionals alike. More Apple owners have learned 6502 assembly language using LISA than all the other assemblers combined. More tutorial material is available for LISA, including books by D. Fudge, R. Hyde, W. Maurer, and R. Mottola. Randy Hyde's 300-page *Using 6502 Assembly Language* is included in the LISA Ed Pac™.

Next we threw in SPEED/ASM™, a set of 6502 subroutines that make programing in assembly language as easy as BASIC. And for those who want to see how it's done, the SPEED/ASM source listings are also included. We also included the LUD # 1 (Lisa Utility Disk # 1) which includes an extended editor for LISA and a LISA source file listing utility. Finally, we added MAXWELL'S Debugger™ to the LISA Ed Pac. This ultra-powerful debugger/monitor makes learning and debugging 6502 assembly language a breeze.

LISA Ed Pac Price $149.95, A $229.75 Value (suggested retail)
Available at dealers everywhere, or directly from:

LAZERWARE

For a copy of Lazwerware's *A Guide to Purchasing a 6502 Assembler for Your Apple II or Apple //e* write us at Lazerware, 925 Lorna St., Corona, CA 91720 or call us at (714) 735-1041
Note: LISA, LISA v2.6, LISA Ed Pac, LISA Educational Package, Speed/ASM, Maxwell's Debugger and LUD are trademarks of Lazerware.
Apple, Apple II, and Apple //e are trademarks of Apple Computer, Inc.

MICRO

for the **Serious Computerist**

MICRO
P.O. Box 6502
Chelmsford, MA 01824
617/256-3649

MICRO is published monthly by:
MICRO, Chelmsford, MA 01824.
Second Class postage paid at:
Chelmsford, MA 01824 and additional mailing offices.
USPS Publication Number: 483470.
ISSN: 0271-9002.
Send subscriptions, change of address, USPS Form 3579, requests for back issues and all othe fulfillment questions to:
MICRO
P.O.Box 6502
Chelmsford, MA 01824
or call 617/256-3649.
Subscription Rates: (per year):
U.S. $24.00 or $42.00 for two years
Foreign surface mail: $27.00
Air mail: Europe $42.00
Mexico, Central America, Middle East, North Africa, Central Africa $48.00
South America, South Africa, Far East, Australia, New Zealand $72.00

Features

Departments

COMPUTER MAIL ORDER

NEC PRINTERS
NEC 2050 $999.00
NEC 3550 $1679.00

PERCOM/TANDOM DISK DRIVES
5¼" 320K Floppy $229.00
5 Meg Hard w/Controller CALL
10 Meg Hard w/Controller CALL
15 Meg Hard w/Controller CALL
20 Meg Hard w/Controller CALL

AMDEK
310A Amber Monitor $169.00
DXY 100 Plotter $599.00
Color II $399.00

AST RESEARCH
Six Pak Plus..from $279.00
Combo Plus II..from $279.00
Mega Plus..from $309.00
I/O Plus..from $139.00

QUADRAM
Quadlink $479.00
Quadboard..as low as $289.00
Quad 512 Plus..as low as $249.00
Quadcolor..as low as $219.00
Chronograph $89.00
Parallel Interface Board $89.00
64K RAM Chips Kit $59.00

MICROPRO
WordStar/MailMerge $349.00
InfoStar $299.00
SpellStar $159.00
CalcStar $99.00

MICROSTUF
Crosstalk $105.00

MICROSOFT
Multiplan $159.00

ASHTON-TATE
dBASE II $389.00
Friday! $185.00

IUS
EasyWriter II $249.00
EasySpeller $119.00
EasyFiler $229.00

CONTINENTAL SOFTWARE
1st Class Mail/Form Letter $79.00
Home Accountant Plus $88.00

IBM

VISICORP

	IBM	APPLE
VisiCalc		$159.00
VisiCalc 4	$159.00	
VisiCalc Advanced		$269.00
VisiWord/Spell	$249.00	
Visitrend/Plot	$199.00	$199.00
VisiLink		$169.00
VisiFile	$199.00	$169.00
VisiSchedule	$199.00	$199.00
Visidex		$159.00
VisiPlot		$135.00
VisiTerm		$75.00
Desktop Plan	$199.00	$169.00
Bus Forecast Model	$75.00	$75.00
StretchCalc	$75.00	$75.00
VisiTutor Calc	$59.00	$59.00
VisiTutor Advanced	$75.00	$75.00
VisiTutor Word	$259.00	$59.00
Visi—On Calc	$289.00	
Visi—On Graph	$179.00	
Visi—On Word	$275.00	
Visi—On Mouse	$159.00	
Visi—On Host	$319.00	

pfs

	IBM	APPLE
Write	$89.00	$79.00
Graph	$89.00	$79.00
Report	$79.00	$79.00
File	$89.00	$79.00
Solutions* as low as	$16.00	$16.00

*Call On Titles

MONOGRAM
Dollars and Sense $109.00

LOTUS
1-2-3 $399.00

PROFESSIONAL SOFTWARE
PC Plus/The Boss $349.00

SYNAPSE
File Manager $89.00

SOFTWARE ARTS
TK Solver $189.00

MBC-550 CALL
MBC-555 CALL

MBC 1100 $1499.00
FDD 3200-320K Drive $389.00
MBC 1150 $1899.00
MBC 1200 $1849.00
FDD 6400-640K Drive $469.00
MBC 1250 $2099.00
PR 5500 Printer $599.00

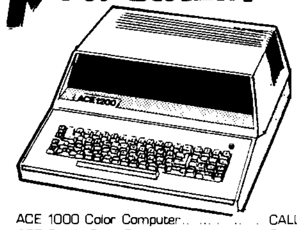
SANYO

APPLE/FRANKLIN DISK DRIVES
MICRO-SCI
A2 $219.00
A40 $299.00
A70 $319.00
C2 Controller $79.00
C47 Controller $89.00

RANA
Elite 1 $279.00
Elite 2 $389.00
Elite 3 $569.00

APPLE IIe STARTER PACK
64K Apple IIe, Disk Drive & Controller, 80 Column Card, Monitor II & DOS 3.3
COMPLETE CALL

FRANKLIN



ACE 1000 Color Computer CALL
ACE Family Pack System CALL
ACE PRO PLUS System CALL
ACE 1200 Office Mgmt. System CALL
"Not IIe Expensive"

TERMINALS
914 $569.00
924 $689.00
925 $739.00
950 $929.00
970 $1039.00

COMPUTERS
Teleport Portable CALL
800A $1099.00
802 $2699.00
803 $1949.00
802H $4695.00
806/20 $4999.00
816/40 $9199.00
1602 $3399.00
1603 CALL

TeleVideo

PRINTERS

AXIOM
AT-100 Atari Interface $239.00
GP-100 Parallel Interface $199.00
AT-846 Interface $89.00

BMC
401 Letter Quality $589.00
BX-80 Dot Matrix $269.00

CENTRONICS
122 Parallel $299.00
739-1 Parallel $199.00
739-3 Serial $249.00

C.ITOH
Gorilla Banana $209.00
Prowriter 8510P $379.00
Prowriter 1550P $599.00
A10 (18 cps) $569.00
8600 P $929.00
F10-40 $999.00
F10-55 $1499.00

COMREX
ComWriter II Letter Quality $499.00

DIABLO
620 Letter Quality $949.00
630 Letter Quality $1749.00

DAISYWRITER
2000 $999.00
Tractor Feed $109.00

EPSON
MX-80FT, MX-100, RX-80, RX-80FT, FX-80, FX-100 CALL

IDS
Prism 80 For Configurations CALL
Prism 32 For Configurations CALL

MANNESMAN TALLY
160L $589.00
180L $799.00
Spirit 80 $309.00

NEC
2010/2030 $899.00
8023 Dot Matrix $379.00
8025 Dot Matrix $669.00
3510 Serial/Letter Quality $1449.00
3530 Parallel/Letter Quality $1499.00
7710/7730 Serial/Parallel $1949.00

OKIDATA
82, 83, 84, 92, 93, 2350, 2410 CALL

SMITH CORONA
TP-2 $399.00
Tractor Feed $119.00

SILVER REED
500 Letter Quality $469.00
550 Letter Quality $699.00

STAR
Gemini 10X $299.00
Gemini P15X $399.00
Delta 10 $559.00
Serial Board $75.00

TOSHIBA
1350 $1699.00

TRANSTAR
120P $499.00
130P $689.00
315 Color $499.00

APPLE INTERFACE CARDS & BUFFERS
Choose from PKASO, Orange Micro, MPC, MicroMax, Tymac, Quadram & Practical Peripherals **PRINTER CABLES** are available for most all computers on the market. We supply all your computer needs.

PAPER SUPPLIES
1000 shts. 8½x11 Tractor Paper $19.99
1000 shts. 14½x11 Tractor Paper $24.99
1 or 2" Address Labels $9.99

MODEMS

ANCHOR
Volksmodem $59.00
Mark IL $79.00
Mark VII (Auto Ans./Auto Dial) $119.00
Mark XII (1200 Baud) $299.00
TRS-80 Color Computer $99.00
9 Volt Power Supply $9.00

HAYES
Smartmodem 300 $209.00
Smartmodem 1200 $499.00
Smartmodem 1200B $449.00
Micromodem IIe $269.00
Micromodem 100 $299.00
Smart Com II $89.00
Chronograph $199.00

NOVATION
J-Cat $99.99
SmartCat 103 $179.00
SmartCat 103/212 $399.00
AutoCat $219.00
212 AutoCat $549.00
Apple Cat II $249.00
212 Apple Cat $569.00
Apple Cat 212 Upgrade $309.00
Cat $139.99

ZENITH
ZT-1 $339.00
ZT-10 $309.00
ZT-11 $369.00

MONITORS

AMDEK
300 Green $149.00
300 Amber $159.00
310 Amber $169.00
Color 1 $279.00
Color 1 Plus $299.00
Color 2 $399.00
Color 2 Plus $419.00
Color 3 $349.00
Color 4 $699.00

BMC
12" Green $88.99
12" Green Hi-Res $119.99
9191-13" Color $249.00

GORILLA
12" Green $88.99
12" Amber $95.99

NEC
JB 1260 Green $109.00
JB 1201 Green $149.99
JB 1205 Amber $159.99
JC 1215 Color $269.00
JC 1216 RGB $429.00
JC 1460 Color $359.00

PRINCETON GRAPHICS
HX-12 RGB $519.00

SAKATA
100 $269.00

TAXAN
210 Color RGB $299.00
400 Med-Res RGB $319.00
415 Hi-Res RGB $439.00
420 Hi-Res RGB (IBM) $489.00
100 12" Green $125.00
105 12" Amber $135.00

USI
Pi 1, 9" Green $99.99
Pi 2, 12" Green $119.99
Pi 3, 12" Amber $149.99
Pi 4, 9" Amber $139.99
1400 Color $269.99

QUADRAM
Quadchrome 8400 $549.00

ZENITH
ZVM 122 Amber $109.00
ZVM 123 Green $89.99
ZVM 135 Color/RGB $469.99

east
800-233-8950
In PA call (717)327-9575, Dept. 40515
Order Status Number: 327-9576
Customer Service Number: 327-1450
477 E. 3rd St., Williamsport, PA 17701

canada
Ontario/Quebec
800-268-3974
Other Provinces 800-268-4559
In Toronto call (416)828-0866, Dept. 40515
Order Status Number: 828-0866
2505 Dunwin Drive, Unit 3B
Mississauga, Ontario, Canada L5L1T1

west
800-648-3311
In NV call (702)588-5654, Dept. 40515
Order Status Number: 588-5654
P.O.Box 6689
Stateline, NV 89449

MasterCard

No risk, no deposit on C.O.D. orders and no waiting period for certified checks or money orders. Add 3% (minimum $5) shipping and handling on all orders. Larger shipments may require additional charges. NV and PA residents add sales tax. All items subject to availability and price change. Call today for our catalog.

COMPUTER MAIL ORDER

Koala Technologies Corporation

KOALA PADS

Atari (Disk)	$75.00
Atari (ROM)	$82.00
C-64 (Disk)	$75.00
C-64 (ROM)	$82.00
IBM	$95.00
Apple/Franklin	$85.00

KOALA SOFTWARE.......CALL

commodore

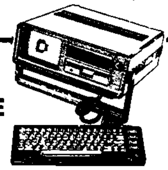
CBM 8032....................$599
CBM 4032....................$599

MSD SD1 Disk Drive	$349.00
MSD SD2 Disk Drive	$599.00
CBM 8096	$869.00
CBM 9000	$999.00
8128-80	$769.00
8032 to 9000 Upgrade	$269.00
2031 LP Disk Drive	$299.00
8050 Disk Drive	$949.00
8250 Disk Drive	$1199.00
4023 Printer	$379.00
8023 Printer	$569.00
6400 Printer	$1399.00
Z—RAM	$499.00
Silicon Office	$699.00
The Manager	$199.00
SoftROM	$125.00
VisiCalc	$159.00

PROFESSIONAL SOFTWARE

Word Pro 2 Plus	$159.00
Word Pro 3 Plus	$189.00
Word Pro 4 Plus/5 Plus...each	$279.00
InfoPro	$179.00
Administrator	$399.00
Power	$79.00

SX-64 PORTABLE $839



VIC 20........................CALL
CBM 64........................$199

C1541 Disk Drive	$249.00
C1530 Datasette	$69.00
C1520 Color Printer/Plotter	$129.00
M-801 Dot Matrix Printer	$219.00
C1526 Dot Matrix/Serial	$299.00
C1702 Color Monitor	$249.00
C1311 Joystick	$4.99
C1312 Paddles	$11.99
C1600 VIC Modem	$59.00
C1650 Auto Modem	$89.00
Logo 64	$49.00
Pilot 64	$39.00
Word Pro 64 Plus	$59.00
Parallel Printer Interface	$49.00
Calc Result 64	$65.00
Calc Result Easy	$39.00
Codewriter 64	$75.00
Quick Brown Fox	$49.00
MCS 801 Color Printer	$499.00
DPS 1101 Daisy Printer	$459.00
Magic Voice Speech Module	$54.00
Desk Organizer Lock	$49.00
Vidtex Telecommunications	$34.95

We stock a full inventory of software for Commodore, such as: Artworx, Broderbund, Commercial Data, Creative Software, EPYX, HES, MicroSpec, Nufekop, Romox, Sirius, Synapse, Thorn EMI, Tronix, UMI, Victory, Spinnaker, Rainbow & Timeworks!

INFOCOM

Zork 1,2,3 (AT/AP/CBM/IBM)	$29.00
Deadline (AT/AP/CBM/IBM)	$32.95
Enchanter (AT/AP/CBM/IBM)	$32.95
Planetfall (AT/AP/CBM/IBM)	$32.95
Witness (AT/AP/CBM/IBM)	$32.95
Starcross (AT/AP/CBM/IBM)	$29.00

ATARISOFT

	IBM/AP	C64/VIC
PacMan	$29.99	$37.99
Centipede	$29.99	$37.99
Dig Dug	$29.99	$37.99
Donkey Kong	$29.99	$37.99
Defender	$29.99	$37.99
Robotron	$29.99	$37.99
Star Gate	$29.99	$37.99

CMO'S PORTABLE CORNER

HP 71B $499.99

41CV........................$199.99
41CX........................$249.99

HP 10C	$51.99
HP 11C	$69.99
HP 12C	$88.99
HP 15C	$88.99
HP 16C	$88.99
HP 75C	$749.99
HPIL Module	$98.99
HPIL Cassette or Printer	$359.99
Card Reader	$143.99
Extended Function Module	$63.99
Time Module	$63.99

TIMEX/SINCLAIR

Timex/Sinclair 1000	CALL
Timex/Sinclair 2086	CALL
16K Memory	$25.00
2040 Printer	$99.99
VuCalc	$17.99
Mindware Printer	$99.99

PC-8201 $599

NEC

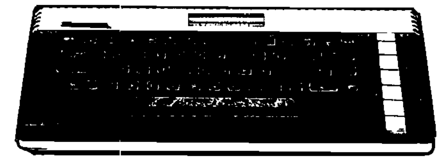
PC-8221A Thermal Printer	$149.99
PC-8281A Data Recorder	$99.00
PC-8201-06 8K RAM Chips	$105.00
PC-8206A 32K RAM Cartridge	$329.00

SHARP

PC-1500A.........$165.99
PC-1250A...........$88.99

CE-125 Printer/Cassette	$128.99
CE-150 Color Printer/Cassette	$171.99
CE-155 8K RAM	$93.99
CE-161 16K RAM	$134.99
CE-500 ROM Library	$29.99

ATARI HOME COMPUTERS



WHILE SUPPLIES LAST!
ATARI 600XL
$149

1010 Recorder	$74.00
1020 Color Printer	$249.00
1025 Dot Matrix Printer	$349.00
1027 Letter Quality	$309.00
1030 Direct Connect Modem	$119.00
1050 Disk Drive	$339.00
CX30 Paddle	$12.00
CX40 Joystick...each	$8.00
CX77 Touch Tablet	$64.00
CX80 Trak Ball	$48.00
CX85 Keypad	$105.00
488 Communicator II	$229.00
4003 Assorted Education	$47.00
4011 Star Raiders	$33.00
4012 Missile Command	$29.00
4013 Asteroids	$29.00
5049 VisiCalc	$159.00
7079 Logo	$79.00
7101 Entertainer	$69.00
7102 Arcade Champ	$75.00
8026 Dig Dug	$33.00
8030 E.T. Phone Home	$33.00
8031 Donkey Kong	$39.00
8033 Robotron	$35.00
8034 Pole Position	$39.00
8036 Atari Writer	$79.00
8040 Donkey Kong, Jr	$39.00
8043 Ms. Pacman	$39.00
8044 Joust	$39.00

DISKETTES

MAXELL

5¼" MD-1	$29.00
5¼" MD-2	$39.00
8" FD-1 (SS/DD)	$39.00
8" FD-2 (DS/DD)	$49.00

VERBATIM

5¼" SS/DD	$26.99
5¼" DS/DD	$36.99

ELEPHANT

5¼" SS/SD	$18.49
5¼" SS/DD	$22.99
5¼" DS/DD	$28.99

HEAD

5¼" Disk Head Cleaner	$14.99

DISK HOLDERS

INNOVATIVE CONCEPTS

Flip-n-File 10	$3.99
Flip-n-File 50	$17.99
Flip-n-File (400/800 ROM)Holder	$17.99

LJK

Atari Letter Perfect Disk(40/80)	$79.99
Atari Letter Perfect ROM(40 col)	$79.99
Atari Letter Perfect ROM(80 col)	$79.99
Atari Data Perfect ROM (80 col)	$79.99
Atari Spell Perfect Disk	$59.99
Atari Utility/MailMerge	$21.00
Apple Letter Perfect	$99.00
Apple Data Perfect	$75.00
Apple LJK Utility	$21.00
Apple Lower Case Generator	$19.00

800XL.................$299
1200XL..............CALL
1400XL..............CALL

PERCOM

AT 88-S1	$369.00
AT 88-A1	$259.00
AT 88-S1 PD	$449.00
AT 88-DDA	$119.00
RFD 40-S1	$449.00
RFD 40-A1	$269.00
RFD 40-S2	$699.00
RFD 44-S1	$539.00
RFD 44-S2	$869.00

TEXAS INSTRUMENTS

TX 99-S1	$279.00

RANA

1000	$329.00

TRAK

AT-D2	$389.00

INDUS

GT-Drive	$379.00

MEMORY BOARDS

Axlon 32K	$59.00
Axlon 48K	$99.00
Axlon 128K	$299.00
Intec 32K	$59.00
Intec 48K	$84.00
Intec 64K	$99.00
Intec Real Time Clock	$29.00

ALIEN VOICE BOX

Atari	$119.00
Apple	$149.00

CONTROLLERS & JOYSTICKS

WICO

Joystick	$21.99
3-way Joystick	$22.99
Famous Red Ball	$23.99
Power Grip	$21.99
BOSS Joystick	$17.99
ATARI/VIC Trak Ball	$34.99
Apple Trak Ball	$54.99
Apple Adapter	$15.99
Apple Analog	$37.99

KRAFT

Atari Single Fire	$12.99
Atari Switch Hitter	$15.99
Apple Paddles	$34.99
IBM Paddles	$34.99
IBM Joystick	$46.99

AMIGA

3100 Single	$13.99
3101 Pair	$19.99
Joyboard	$37.99

TG

Atari Trak Ball	$47.99
Apple Joystick	$47.99
Apple Trak Ball	$47.99

east
800-233-8950

In PA call (717)327-9575, Dept. 40515
Order Status Number: 327-9576
Customer Service Number: 327-1450
477 E. 3rd St., Williamsport, PA 17701

canada
Ontario/Quebec
800-268-3974

Other Provinces 800-268-4559
In Toronto call (416)828-0866, Dept 40515
Order Status Number: 828-0866
2505 Dunwin Drive, Unit 3B
Mississauga, Ontario, Canada L5L1T1

west
800-648-3311

In NV call (702)588-5654, Dept 40515
Order Status Number: 588-5654
P.O.Box 6689
Stateline, NV 89449

VISA

CANADIAN ORDERS: All prices are subject to shipping, tax and currency fluctuations. Call for exact pricing in Canada.
INTERNATIONAL ORDERS: All shipments outside the Continental United States must be pre-paid by certified check only. Include 3% (minimum $5) shipping and handling.
EDUCATIONAL DISCOUNTS: Additional discounts are available to qualified Educational Institutions.
APO & FPO: Add 3% (minimum $5) shipping and handling.

MICRO™ Editorial

Once Upon a MICRO

Once upon a time, MICRO began as a magazine to promote the 6502 microprocessor. At that time, back in the murky mists of microcomputing, 1977, no one was giving this marvelous chip any attention. You could read many issues of Byte without even encountering it. We felt that this chip, and the KIM-1 microcomputer that MOS Technology had produced to demonstrate the abilities of the 6502, deserved better treatment. The rapid growth of MICRO showed that we were right!

Once upon a time, MICRO was more a 'community' of 6502 users than it was a 'publication'. MICRO's readers were willing to tackle the new micros, solve the many problems that were encountered, and share their information with other readers. It was an exciting time of exploration and experimentation. Many important features were discovered, problems solved and projects generated by the MICRO reader/author.

Once upon a time, MICRO helped lead its readers into new areas by systematically exposing them to other microcomputers, microprocessors, languages, techniques, hardware projects, and so forth.

Once upon a time, MICRO provided very rapid turn-around on material submitted for publication. Articles were typically published within two or three months of initial receipt. This rapid turn-around was satisfying to the authors and useful to the readers.

Once upon a time, MICRO was a small, over-worked but happy staff that took pride in producing a top quality product.

Once upon a time, MICRO was directed by an individual who had experience in software - from operating system design through applications, and hardware knowledge - from simple interfacing up to designing a complete disk-oriented microcomputer system.

Once upon a time, MICRO provided an up-to-date catalog of important hardware and software products, in a standardized format that made it easy to use.

Once upon a time, MICRO had a panel of expert reviewers who provided accurate, unbiased, and timely reviews of new products.

Well, "Once Upon A Time" is now! While MICRO has tried a lot of different ideas, particularly during the past year, it has now returned to its 'roots'. We have worked hard to get MICRO back on track as the premier magazine for people who are serious about all aspects of the 6502/6809/68000 family of microcomputers. Some of the obvious changes have included moving MICRO back to Chelmsford, MA where it began, my reassuming the **active** role of Publisher and Editor-in-Chief, and numerous changes in the staff to streamline and improve our overall operation. In the past few months we have cleared the queue of all out-of-date articles and reviews that had been accumulated, have re-established active dialog with many key authors, have worked out internal procedures to insure rapid response to all submissions, have developed improved listing methods for both assembly and BASIC listings, and much more. Other changes are underway, some of which will take time to develop, all of which are aimed at making MICRO work for you.

One new way in which MICRO will work is to present material on diskette. Many program/articles are received that are '**too long**' to print or to key in but are '**too good**' not to use. Rather than ignore this significant material, or hold it for an eventual book/disk, MICRO will now offer certain materials on disk. See the announcement on page 80 of this issue for details on our first offerings.

If you are an author, MICRO guarantees that your manuscript will be reviewed and you will receive notification within two weeks of receipt. This rapid response will serve to get your material into print quickly with prompt payment, and will insure that the MICRO readers are getting the most current information.

If you are a reader, MICRO invites you to become a more active participant in the world of microcomputing. Tell us, through the June Reader Survey, what you want MICRO to do for you. Send us your ideas, suggestions, feedback. (We do listen! The negative reader feedback that we received on our 'new, improved' listing techniques in the November and December issues made us find better methods.) And, most of all, write articles to share your knowledge and understanding with others. In this fantastic world of microcomputers, nobody knows everything, and everyone knows something.

MICRO is your magazine. Make it work for you.

Robert M. Tripp

Editor-in-Chief

This Month's Cover

When he looked at this windmill, Don Quixote saw double. He fought an imaginary giant, and he lost.

In *Double Vision*, Valerie and Alan Floeter fight a real giant ... the problem of one long listing using the CATALOG routine. Unlike Don Quixote, the Floeters win their battle. Now your listing can be condensed into two, three, or four columns, but don't worry—your not imagining it—you've got Double Vision.

Letterbox

Dear MICRO:

Some of your recent issues have had parallel articles for the various computers you cover. I like this feature. In fact, it is the main reason I renewed my subscription. It is frustrating to see neat programs written for other computers, but not for mine.

Consequently, I was very disappointed that the well-written article on *Fast Low Cost A/D Converter*, MICRO 69, did not have listings for use of the converter with the Atari Computers. No A/D converters are available for the Atari (to my knowledge). This could have been very useful. The Atari was also slighted in the **Adding Computer Senses to Your Micro**.

Your excellent **Interface Clinic** suffers from the same problem. I should think that there are other Atari users who might also wish to have routines useful for interfacing the Atari with analog circuitry.

I hope you can extend this type of article to include the Atari in the future. Thank you for your consideration.

Michael Soso
Seattle, WA

Your points are well taken. The **generality** *problem discussed above is even more prevalent when dealing with the Atari. The BASIC used in the Atari is somewhat unique. While there are many minor differences between the BASICs on the Apple, Commodore and Color Computer – they do have a lot in common. Atari is sort of* out-in-left-field. *For this issue, for example, I went to generalize the* **Talking to your Printer**.
First I had to wade through *the Atari* **OPEN** *and* **XIO** *commands to setup the input and output, then had to* **DIM**ension *all of the string variables, and then realized that due to the strange way Atari BASIC handles string concatenation – there was no way to get the program to work!!! Talk about frustration!*

Some of the other programs you mentioned could have been, and probably should have been, converted. If any Atarist has converted them, we would be happy to print updates. Let me make two proposals for future articles. First, MICRO will make a greater effort to perform Atari conversions where possible/practical. Second, if any Atari readers are interested in performing such conversions, we will work with them, pay a modest remuneration, and provide program/projects to convert. We can not do it all, but we can all do it together.

Dear Editor,

I read your editorial "Is It Reasonable?" in February 1984 MICRO #69 with interest. Many of your thoughts and statements are true and I agreed with them. I do think you touched on a very important aspect of the APPLE success (and failure). That is "Third Party" vendors, which I'll come back to in a moment.

I believe the success of the microcomputer in the home and workplace stems from the fact that we are trained to use tools of "convenience", for example, log tables, slide rule, electronic calculator and the microcomputer. We must have some knowledge of their function to use them successfully and effectively. It is exciting to watch a human float effortlessly in space, but the thoughts of the details of what it really took to put him there and get him back are much more exciting. Some knowledge of the intricate steps required is where it is really at. So it is with the microcomputer.

What Apple, Inc. did with the Apple II was give the curious the opportunity to learn the intricate details which cause the II to function. They produced a Disk Operating System and Monitor that was easily and quickly understood. Your publication published a complete understanding of the Apple II's operating system. I think the way the 'Steve(s)' started made this environment necessary. They needed the support of the Home Brew Computer Clubs, and also of Third Party Vendors. If the software didn't get written and published as fast as it did, I believe Apple wouldn't be as successful as it is. The resources weren't available at the time. It is the software vendors and publications such as yours that contributed a measureable amount of resources to Apple's success.

It seems to me the Apple III, Lisa, MAC, et al, will follow the path of the TI-99 unless Apple will facilitate easier learning of the Operating System. The person in the business place may only want applications software, but there are orders of magnitude more at home wanting to write their own software hoping it will be of sufficient quality to be published. Apple needs to loosen their management philosophies regarding the MAC and provide a simpler operating system.

I am a co-founder of a 170 member Apple users group here in Silicon Valley. We were fortunate to have Apple's sales department demonstrate the MAC the day after it was unveiled. We had 300 people attend the presentation - an exciting turn out! Disappointment quickly set in when I discovered the complexity of the operating system. I decided the MAC was not something I would be interested in. I think MAC will follow Lisa unless Apple wakes up and provides a much simpler operating system so the Third Party vendors can contribute again.

I am now waiting for the new 650XX chip that has been reported in various news releases. If it doesn't satisfy the simple Apple II operating system concept with much expanded memory, I will seriously look at the Saybrook or QWERTY system again. I suppose you have guessed what I think is reasonable. It is a source of hardware and information that will give our tools of convenience a chance to be even more so.

Robert C. Madden
San Jose, CA

Sir

"The Apple/// had limited capabilities..."!!??

Surely you jest! 256K of pure RAM: what could be more versatile? An operating system which BYTE magazine called "the most

sophisticated operating system available for an 8-bit machine". The ability to run virtually all Apple][software. Can you seriously consider this to be limited capability?

Now, there is no doubt that the /// had some early hardware problems, which in turn discouraged software developers from jumping on the bandwagon. This, rather than "limited capability" resulted in the lack of early acceptance of the ///. Apple has now revitalized the ///, set up specific resources for it, and even published a booklet (approximately entitled '*Will someone tell me what you can do with an Apple ///?*') which lists a great range of software available for the ///.

I develop software on the /// for both it and the][. I also use the /// for all my other work: word processing, data base, modem, Pascal, BASIC, Assembly ... etc. I will gladly stack up the capabilities of the Apple /// against any other 8-bit machine on the market, regardless of price or manufacturer (and even against some of the pseudo-16s). Would you care to enumerate its limitations?

Tracy Valleau
Pacific Grove, CA

I personally was unimpressed with the Apple /// when I saw it. I was at that time completing development of a 6809 – based system that sold for $500 less than the basic Apple /// and offered almost eight times the disk capacity, had a far superior keyboard, included many hardware features and a complete package of user – friendly software. I really had expected a lot more from Apple for the price – not just more memory. I guess today I would say it is perhaps a good computer, certainly not a great one. The next writer provides another possible reason for the lukewarm reception the Apple /// got.

Dear Sirs:

Although this letter will refer to the program by Joseph Kattan in **MICRO 71**, my criticisms are really directed at MICRO's editorial policies, rather than the specific program. The Credit Register program looks like a good idea, and I would like to run it on my computer. However, it is written in such a way that it is essentially not transferable to any computer other than the Atari. The GRAPHICS commands, as also the PEEKs and POKEs are totally specific to the Atari, and lacking REMs as to their function, it is impossible to reproduce them on another machine. If the program involved something that had to be hardware dependent, such as a hi-res graphics presentation, there might be excuse for this. However, the screen presentation that is shown looks like fairly straightforward printing, which it should be possible to generate with standard PRINT statements.

You are in a favorable position for insisting on some kind of **standard** BASIC in your program listings, to improve as far as possible the portability of programs from one computer to another. If this means that all programs are restricted to a minimum implementation of BASIC (a sort of *lowest common denominator*), this is not necessarily a bad thing. It is very elegant to use all possible bells and whistles that are specific to your computer, in order to get the most sophisticated display; but if this is only achieved at the expense of portability, I believe it is a bad bargin.

Rolf B. Johannesen
Rockville, MD

I agree with everything that you say, except for the problem being one of 'MICRO's editorial policies'! We evaluate every article with machine **generality** *in mind. Literally hundreds of programs/articles have been rejected because they were limited to a single computer. The ideal would be for every program to work on every machine. Unfortunately, there are a number of factors working against this ideal. First, most authors have expertise on one micro and are often not aware of what is specific to their BASIC. Second, it is easier to write machine specific BASIC. Third, most authors do not have multiple micros for testing various versions. Fourth, it takes a great deal of work to take a program that has not been written with generality in mind and generalize it. We have worked many hours recently just to generalize a few programs, including* **Smart Modem**, *(converted for three additional microcomputers),* MICRO 68; **Adding Computer Senses**, *(converted for two additional microcomputers),*MICRO 69; **Least-Squares Curve Fitter** *and* **PEEKing TOM**, MICRO 70; **Talking to Your Printer**, *in this issue; and so forth. Fifth, a* **generalized** *program requires testing. It may require hardware configurations that neither the author nor MICRO possess. Sixth, errors may be induced into the program during the generalization process.*

The **Talking to Your Printer** *article shows one technique that we use and hope that other programmers will adopt where possible. We plan to describe other techniques in future issues. Unfortunately we are not in a position to* **insist**: *we* **request**, *and we* **conjole**, *and we do* **reject**.

Dear Sir,

I haven't been able to enter the program *Master Directory for the Apple*, **MICRO 67/69** into my Apple II+ with Language Card. My usual procedure with Machine Language programs is to convert to a Hex dump but the Master Directory listing is one that I have never seen before and it does not seem suitable to this method.

Would you please advise how I could go about using this program.

Herman F. Schulz
Schenectady, NY

Mr. Hill's program was **too long** *to print and* **too good** *to ignore! We normally print the object code along with the source. In this case we dropped the object code to save space. Due to a number of letters and phone calls, we planned to print the hex dump in this issue. Turns out it would take a full four pages, four columns per page! What to do? We have arrived at two solutions: 1. We will provide a complete listing including the object code for $1.00 to cover copying and a self – addressed, stamped – envelope; or, 2. We will provide the assembler source (in LISA format) and the binary file (BRUN format) on a diskette, and the printed listing, for $15.00. See the* **MICRO Diskette Service** *announcement on page 80*

Reviews in Brief

Product Name: **IDS, An Integrated Development System for the Apple II Plus**
Equip. Req'd: Apple II or II+
Price: $85.00
Manufacturer: R.R. Michaels, Inc.,
Box 565
Leesburg, VA, 22075
703/777-1933

Description: A utility package, written in 6502 machine langauge, to support Applesoft programming. Allows for the easy construction of screen displays for data entry, file structures for record storage and retrieval of output formatting for reports.

Pluses: The package is easy to use. It includes an editor for constructing display screens, which are keyed to variable labels; this permits the Applesoft programmer to coordinate their variables directly with the input display screen. Input edit checks can be performed to reduce the chance of key-stroke error. The record definition system permits easy storage and retrieval of both sequential and randomly accessed files. Individual fields, as well as entire records, may be stored or retrieved. One Applesoft weakness is in the output of data, where it lacks a PRINT USING capability. IDS permits the programmer a variety of display formatting, including specified decimal places, embedded commas and dollar signs and right justification. The IDS system uses CALLS to reserved variable names which perform each function (all of the calls begin with Z to avoid confusion with other BASIC variables).

Minuses: The IDS package has a modest RAMN overhead of a minimum of $2000 bytes. In addition, several of the structure definition tables reside just below this address. Thus, the user will sacrifice at least 8K to use the IDS software, although many Applesoft routines are eliminated, shortening the space needed for BASIC code.

Documentation: The manual is well written. It begins with a tutorial on the use of the IDS routines, including the construction and use of each of the three subsystems. A reference section describes each command available.

Skill level: The package will be of most interest to the Applesoft programmer writing commercial grade software.

Reviewer: David Morganstein

Product Name: **Magic Memory**
Equip. Req'd: Apple II+, IIe with one disk drive and 48K RAM
Price: $100
Manufacturer: ARTSCI
5547 Satsuma Blvd.
North Hollywood, CA 91601

Description: Magic Memory bills itself as an electronic address book. It is, in fact, a flexible way to create and recall a variety of information. All entries can be cross-indexed, easily updated, and printed. Files can be saved on any disk, making it virtually impossible to run out of storage space.

Pluses: Looking to future developments, the program is entirely compatible with a hard disk drive and disk space is reserved within a submodule for new utilities that may be created.

Minuses: The copy-protected master disk cannot be copied to a hard drive and the 70-column video driver can only be used with a 64K system. Memory does not let you carry a file format from one file to another; rather you start each file blank and have to enter all data.

Documentation: A looseleaf manual provides ample instructions and some technical information.

Skill level: Intermediate to advanced.

Reviewer: Mike Cherry

Product Name: **Super Text**
Equip. Req'd: Commodore 64 with one disk drive and a printer
Price: $100
Manufacturer: Muse Software
347 N. Charles Street
Baltimore, MD 21201

Description: A word-processor with a software-based 80 column display, Super-Text is loaded with features: creating/saving files, block moves, justification, automatic page numbering, find and replace, tabs, and imbedded control characters. Also available are file merge, on-screen help, word counting, and "autolinking" your files to the printer.

Pluses: Super-Text provides several printer parameters which can be adapted to fit almost any printer and interface., The 80 column display is a "bonus" feature and does not gobble up all your memory.

Minuses: Creating and editing text occur in separate modes. Jumping from mode to mode will slow you down and confuse you at first. Also, the screen will not necessarily show the printer's format. You will need to refer to a preview section to verify the printout is the way you want it.

Documentation: A spiral booklet contains tutorial and technical information.

Skill level: Beginner and up.

Reviewer: Mike Cherry

Product Name:	**Computer Mechanic**
Equip. Req'd:	Commodore 64 with disk drive or cassette
Price:	$60
Manufacturer:	Softsync, Inc. 14 East 34th Street New York, NY 10016

Description: A diagnostic program to help pinpoint mechanical problems with your car. Mechanic also teaches the basics of car maintenance and sets up a repair history and maintenance schedule for any car.

Pluses: Mechanic will prepare a standard disk to accept files giving you room for hundreds of records. The use of the Commodore's graphics and color abilities is excellent and the advice is sound and helpful.

Minuses: Error-handling is marginal. Data entries are not adequately checked for proper input and error messages may confuse the beginner. Mechanic's simple approach limits the diagnostic advice/record-keeping to an introductory level.

Documentation: A thin 6-page pamphlet provides orientation but no technical information.

Skill level: Intermediate to advanced level. Poor error-handling means a beginner may have trouble with this program.

Reviewer: Mike Cherry

Product Name:	**Delta Drawing**
Equip. Req'd:	Apple II, II+, IIe with one disk drive & 48K RAM
Price:	$40
Manufacturer:	Spinnaker Software 215 First Street Cambridge, MA

Description: A FORTH-based program geared towards elementary school use lets you create drawings with simple keyboard commands. Various configurations allow for color fill, background color, preprogrammed patterns, saving and printing programs in text or graphics modes.

Pluses: Easy to learn, Delta Drawing is fast and pleasing. Children will be able to create interesting pictures with only a little practice. The "color fill" command is especially fun to watch.

Minuses: Only the Grappler+ interface is supported for printing graphics; all else will print only text. Patterns saved can not later be recalled and included in BASIC programs.

Documentation: An excellent tutorial and separate 'flash cards" illustrating various patterns are included.

Skill level: Beginner and up.

Reviewer: Mike Cherry

Product Name:	**Mail Controller**
Equip. Req'd:	Commodore 64 with 1541 Disk Drive, 1525 printer or other with interface.
Manufacturer:	Orbyte Software Box 948 Waterbury, CT 06720

Description: An easy to use mail list program. Allows over 2000 entries per disk with editing functions available. Will print labels in one across format or print out on paper stock taking advantage of your paper width. The program allows formatting a new data disk in order to access more records. This makes it possible to have extremely large files across several disks.

Pluses: The program is powerful and easy to use. All work starts at the menu and the function keys are taken advantage of to simplify work. A help screen is provided for the New Disk and Data Entry functions. Mail Controller may also be used as a small database for other than mail lists, although the amount of information storage is limited.

Minuses: There is a limit of 73 characters that may be used for the fields in a standard mail list format. This requires careful field setup.

Documentation: The 38-page manual is one of the easiest to use that I have seen. Each function is made clearly understandable.

Skill level: Anyone, from beginner to expert, would be able to use this program.

Reviewer: Richard E. DeVore

Product Name:	**Experiments in Human Physiology**
Equip. Req'd:	Apple II, II+, IIe
Price:	$249 (Demo disk available for 30-day preview)
Manufacturer:	HRM Software 175 Tompkins Avenue Pleasantville, NY 10570

Description: A combination of hardware and software to implement a variety of experiments in Biology and Human Physiology including: Psychomotor Response Time, Calibration of Temperature Probe, Skin Temperature, Respiration Rate, Heart Rate and Polygraph Testing. A useful supplement ot a High School Biology class. All of the experiments would easily fit into the classroom curriculum, helping the student to further understand basic functions by first hand experiments and encouraging further exploration in this area.

Pluses: The experiments are simple but dramatic, giving the impression of a "mini-laboratory". The students learn by doing, gaining not only class work, but experience in using computers as well.

Minuses: None.

Documentation: Well written and clearly explaining the experiments.

Skill level: Beginner; the hardware connections are clearly explained, although they probably should be done by the teacher.

Reviewer: Edouard Garcia

Product Name:	**Pro-Color-File 2.1**
Equip. Req'd:	TRS-80 Color Computer
Price:	$79.95
Manufacturer:	Derringer Software P.O. Box 5300 Florence, SC 29502

Description: A database utility. File definition capability allows up to 60 fields per record, to a maximum length of 1024 bytes. Fields can be defined as numeric or alphanumeric. Report formatting capabilities include math functions, report layout and definable work fields for use in reports. Up to five distinct report formats may be defined and invokable at any one time. Search, select and sort features are available for databsase manipulation. Data entry is accomplished via quick entry screens designed by the user. Up to 5 separate data entry screens may be defined per logical record.

Pluses: Good flexibility in design allows for a wide variety of applications. Subtotals, totals and averages can be automatically calculated in reports. Other user-definable formulas can add versatility to the report writer feature. The entire program is written in Extended Color Basic and is provided as user-modifiable code. This allows the user to make such things as printer baud rate settings a permanent part of the program. Record segmentation provides the ability to add fields even after records have been entered. Select and sort features are quick and efficient. Special menu format allows for end user input with reduced menu. Password protection is available on selected fields.

Minuses: The program does not provide for boolean operations during report writing aside from the standard selection process. The documentation claims that a field name can be up to 15 bytes long, but the program would allow only a 12 byte name. The program does not provide any automated word procesing capabilities. While this can be accomplished using the report writer, much manual intervention is required during the printing phase.

Documentation: A 35-page manual is well-written and easy to understand. It makes good use of examples and the diskette also includes those same examples to give the new user an established database to practice with.

Skill level: Intermediate. Programming skill is not required, but some familiarity with computer records is useful.

Reviewer: Norman Garrett

Product Name:	**HJL-57 ColorComputer Replacement Keyboard**
Equip. Req'd:	TRS-80 Color Computer
Price:	$79.95
Manufacturer:	HJL Products P.O. Box 24954 Rochester, NY 14624

Description: A direct replacement for the standard Color Computer keyboard, but unlike others on the market, it has the layout and color scheme of the original with the addition of a longer spacebar and four function keys (one locking). It includes installation instructions, necessary hardware and a replacement bezel. The keyboard is fully shielded ahd has the connecting cable installed (the purchaser must specify the computer version desired so that the correct connecting cable can be determined). The keyboard rests at about the same angle as the original, with the overall contour slightly modified. It comes with a one year guarantee.

Pluses: The contour of the keyboard (the slight variation of angle between rows of keys) is modified a bit to give it a more natural feel. This is especially apparent with the spacebar, which is much more accessible than on the original. The texture and sculpture of the low profile keys are significantly improved. An additional benefit is the RFI shielding included, which noticeably reduced the RFI on my television.

Installation is straightforward. For a person who has never opened the computer case, it would probably take a maximum of 30 minutes. The only modification is to shorten one plastic post. The unit rests on the original posts. Manufacturer telephone response to questions is good. The finished appearance is good, blending well with the original and being truly a replacement and not a modification.

Minuses: The effect of the locked PF2 key on other keys needs to be explained (more explanation on the actual use of the function keys in general would be helpful). This would allow a programmer to better utilize the programmed key functions.

Documentation: Consists of excellent, easy to follow installation instructions, the decimal values generated by the function keys, and a sample program which will program your function keys as follows: F1 dumps the current screen to the printer; F2 allows auto repeat of any key (F2 locks); F3 flips between upper and all lower case; F4 acts as a control key and subtracts 64 from the ASCII value of any key.

Skill level: Installation requires no technical experience. The instructions are geared to a non-technical installer.

Reviewer: Norman Garrett

Product Name:	**Flight Simulator II**
Equip. Req'd:	Apple II + 48K, DiskDrive
Price:	$49.99
Manufacturer:	SubLOGIC Corporation 713 Edgebrook Drive Champaign, IL 61820

Description: The long awaited sequel to Flight Simulator I. An incredibly well thought out product of real value to pilots and fascinating to those not aviation minded. The package includes maps of the four areas of the U.S. modeled in detail on the main disk. The company advertises the availability of other scenery disks. Care has been taken to simulate the intricacies of communications and navigation that are in real life the most demanding tasks of a pilot.

Pluses: Sheer attention to detail. User variable weather is a particularly valuable feature in that it brings home to the user precisely how poor weather creates chaos with flying. For the younger user, a World War I dogfight game is included as a special option of the main disk.

Minuses: There really are not many. In places, the instructions could be improved. There is a tendency to assume too much aeronautical knowledge on the part of the user. The authors seemed to be aware of it, but were not entirely successful in avoiding the problem.

Documentation: Overall, I was pleased with it. The manual was printed in a professional manner, and the incredible detail of the product was handled nicely. Each feature is explained without hype or unnecessary enthusiasm.

Skill level: A novice computer user who follows the manual should have no trouble using the product.

Reviewer: Chris Williams

Product Name:	**The World of Counting**
Equip. Req'd:	Apple II + or IIe
Price:	$24.95
Manufacturer:	Educomp Enterprises 191 North 650 East Bountiful, UT 84010

Description: Designed to teach counting principles to learning-disabled class or regular preschool. Provides examples, demonstrations, quiz questions, and a final test, using hires graphics, music and sound effects for reinforcement. Student scores and response times are displayed at end of lesson.

Pluses: Lots of repetition and reinforcement. Program written in Apple Pilot - can be customized to meet user's specific needs. Good graphics.

Minuses: Pictures are drawn very slowly (inherent problem in Pilot). Musical reward may be confused with musical number prompt. Scores are not stored on disk for later reference.

Documentation: 8-page pamphlet with excellent program description and directions. Software also shows instructions.

Skill level: 3 to 6 year old (mental age); adult to start program.

Reviewer: Mary Gasiorowski

Product Name:	**Practicalc II**
Equip. Req'd:	48K Apple II+, IIe and compatible computer
Price:	$69.95
Manufacturer:	Micro Software International Inc. The Silk Mill 44 Oak Street Newton Upper Falls, MA 02164

Description: This is not another Visicalc clone although it does have the same basic features; 80 column width, scrolling, columnar movement/expansion, and all the other spreadsheet "musts." It has some things that the others are lacking, for instance database management that enables you to do alpha and numeric sorting and searching. There are also prompts for entry during calculation and printing of list formulas.

Practicalc II was designed with the nonprofessional user in mind. It certainly is capable of being used in a business setting but unlike most "professional" packages it has some friendlier additions. For example, you may not use a particular spreadsheet but once a month, so chances are the next time around you won't remember how you set things up. This is not a trivial matter when you are dealing with columns and numbers. Practicalc II saves a spreadsheet with the menu that contains all of the printer settings and other pertinent information.

Pluses: Unlike the other spreadsheet packages you have seen, the price for this one is only $69.95! Perhaps you had previously found it hard to justify an expenditure of several hundred dollars for something that you only needed a few times a month. At this price intermittent use is justifiable, particularly when you think of the time saved and the frustration avoided.

One unique and handy "extra" offered by Practicalc II is that it includes a word processing package. It is your basic WP, but has a few nice additions not usually found. One of these is the capability of typing columnarly - newspaper style. This is a feature many major WP packages do not have. It is one of those things that might not be useful to most people on a regular basis, but when you need it - what a blessing! The standards - insert, delete, etc., are nicely implemented. When deleting/inserting in newspaper-style typed text, correcting one column does not affect the other.

Besides the bargain price, Practicalc II has another major difference - it is not copy protected. Microsoftware hopes this feature will not be abused, but instead will aid its customers by allowing them to have a copy of Practicalc II on different diskettes. This certainly would enable more facile use of their product and make life a little easier for the user. Using 15K of memory, there is plenty of room for other things. The actual code can be accessed if you are willing to disassemble it (use Big Mac) and put the whole thing back together. I would suggest you make a few copies for backup first.

Minuses: The one area in which Practicalc II is not as proficient is speed. The difference is minimal, most noticeable when saving a spreadsheet. Because each sheet is saved with a copy of the menu and its settings, it takes a little longer to be stored on disk - a reasonable tradeoff to most users. The scrolling and screen movement are not as fast as its competitors, but again the difference is negligible.

Documentation: The documentation for Practicalc II is clearly written with examples and logical steps for procedures. It contains an Index and a good Table of Contents - both very useful. Future addenda should be available to users for a nominal fee. Also available will be diskettes containing new versions to help kill the bigger bugs. These will sell for $5.00 a diskette.

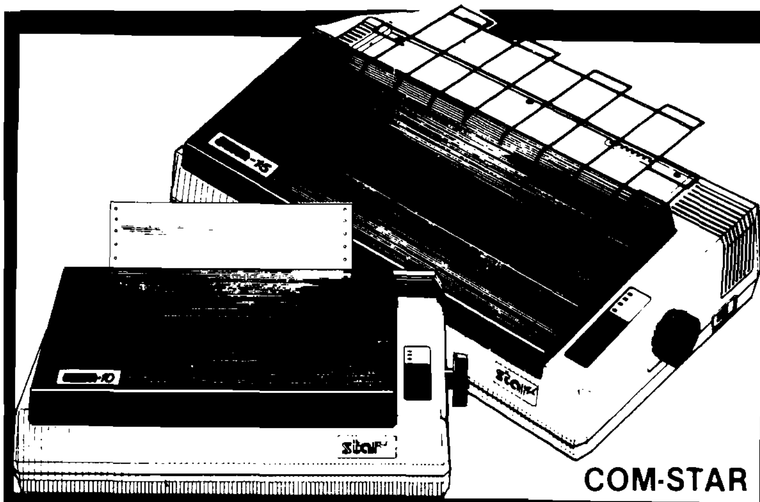
Skill level: The level of expertise needed by the user ranges from beginner to expert. The beginner can learn the basics rather easily; the more advanced the user the more options and features he/she will be able to utilize.

Reviewer: Mark S. Morano

MICRO


$69.00

80 COLUMN PRINTER SALE—$149.00*



COM-STAR T/F

Tractor Friction Printer

only $199**

•15 Day Free Trial -180 Day Immediate Replacement Warranty

• Lowest Priced, Best Quality, Tractor-Friction Printers in the U.S.A.
• Fast 80-120-160 Characters Per Second • 40, 46, 66, 80, 96, 132 Characters Per Line Spacing
• Word Processing • Print Labels, Letters, Graphs and Tables • List Your Programs
• Print Out Data from Modem Services • "The Most Important Accessory for Your Computer"

*STX-80 COLUMN PRINTER—$149.00

Prints full 80 columns. Super silent operation, 60 CPS, prints Hi-resolution graphics and block graphics, expanded character set, exceptionally clear characters, fantastic print quality, uses inexpensive thermal paper! Best thermal printer in the U.S.A.! (Centronics Parallel Interface).

**DELUXE COMSTAR T/F 80 CPS PRINTER—$199.00

The COMSTAR T/F (Tractor Friction) PRINTER is exceptionally versatile. It prints 8½" x 11" standard size single sheet stationary or continuous feed computer paper. Bi-directional, impact dot matrix, 80 CPS, 224 characters. (Centronics Parallel Interface).

Premium Quality—120 CPS COMSTAR T/F SUPER-10X PRINTER—$289.00

COMSTAR T/F (Tractor Friction) SUPER-10X PRINTER gives you all the features of the COMSTAR T/F PRINTER plus a 10" carriage, 120 CPS, 9 x 9 dot matrix with double strike capability for 18 x 18 dot matrix (near letter quality), high resolution bit image (120 x 144 dot matrix), underlining, back spacing, left and right margin settings, true lower decenders with super and subscripts, prints standard, italic, block graphics and special characters, plus 2K of user definable characters! The COMSTAR T/F SUPER-10X PRINTER was Rated No. 1 by "Popular Science Magazine." It gives you print quality and features found on printers costing twice as much!! (Centronics Parallel Interface) (Better than Epson FX 80).

Premium Quality—120 CPS COMSTAR T/F SUPER-15½" PRINTER—$379.00

COMSTAR T/F SUPER 15½" PRINTER has all the features of the COMSTAR T/F SUPER-10X PRINTER plus a 15½" carriage and more powerful electronics components to handle large ledger business forms! (Better than Epson FX 100).

Superior Quality SUPER HIGH SPEED—160 CPS COMSTAR T/F 10" PRINTER—$489.00

SUPER HIGH SPEED COMSTAR T/F (Tractor Friction) PRINTER has all the features of the COMSTAR SUPER-10X PRINTER plus SUPER HIGH SPEED PRINTING—160 CPS, 100% duty cycle, 8K buffer, diverse character fonts, special symbols and true decenders, vertical and horizontal tabs. RED HOT BUSINESS PRINTER at an unbelievable low price!! (Serial or Centronics Parallel Interface)

Superior Quality SUPER HIGH SPEED—160 CPS COMSTAR T/F 15½" PRINTER—$579.00

SUPER HIGH SPEED COMSTAR T/F 15½" PRINTER has all the features of the SUPER HIGH SPEED COMSTAR T/F 10" PRINTER plus a 15½" carriage and more powerful electronics to handle larger ledger business forms! Exclusive bottom paper feed!!

PARALLEL INTERFACES

For VIC-20 and COM-64—$49.00
For All Apple Computers—$79.00

NOTE: Other printer interfaces are available at computer stores!

Double Immediate Replacement Warranty

We have doubled the normal 90 day warranty to 180 days. Therefore if your printer fails within "180 days" from the date of purchase you simply send your printer to us via United Parcel Service, prepaid. We will IMMEDIATELY send you a replacement printer at no charge, prepaid. This warranty, once again, proves that WE LOVE OUR CUSTOMERS!

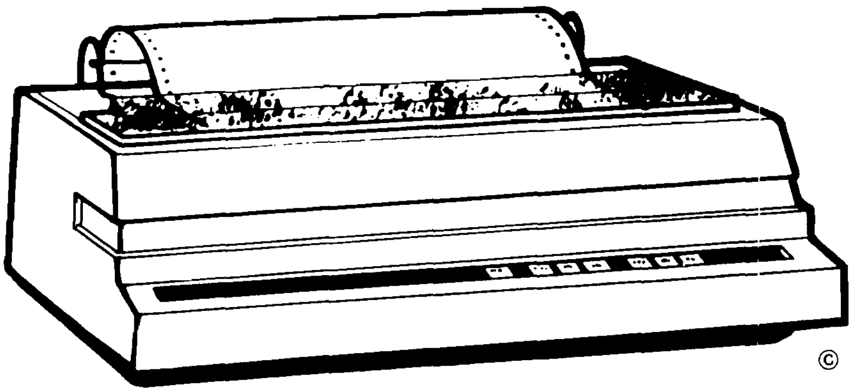
Add $17.50 for shipping, handling and insurance. WE DO NOT EXPORT TO OTHER COUNTRIES EXCEPT CANADA.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail! Canada orders must be in U.S. dollars. VISA — MASTER CARD ACCEPTED. We ship C.O.D.

PROTECTO ENTERPRIZES (WE LOVE OUR CUSTOMERS)
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

SUPER-10" ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890

The Accurate Printer



by Richard Marmon

Have you ever typed a program from a book or magazine into your Atari computer that used graphics characters or inverse characters? Have you been frustrated when you tried to list the program on your Epson (or Epson-compatible) printer only to have the printer go crazy? If you have the Graftrax or Graftrax-plus option for your printer, then Accu-Print will solve this problem for you - and give you some added desirable features as well. With Accu-Print in control of your printing, your Epson printer will faithfully reproduce each and every character that the Atari line of computers can generate on the screen with no exceptions.

What Accu-Print Can Do

Figure 1 shows a little nonsense listing using Atari graphics characters and inverse characters. If you type these statements into your computer (using the appropriate keystrokes to obtain the special characters as described in the Atari Basic manual) and then list them to your printer, you will obtain the results shown in Figures 2, 3, and 4 printed on three sheets of paper. The printer will pretty much garble the listing and seem to go crazy with form feeds as it is printing.

This is because there is an incompatibility between the ATASCII codes used to represent the computer's character set and the ASCII codes used to represent the printer's character set. Some ATASCII values representing inverse characters to the computer represent different Epson-style graphics characters to the printer. In fact, the special Atari graphics and inverse characters just aren't included in the Epson printable character set. And some ATASCII values are interpreted by the printer as special control characters. Hence, the printer form feeds when you don't really want it to. This is quite a messy state of affairs!

I wrote Accu-Print in order to get around these difficulties. It seemed to me that the printer's graphics capabilities could somehow be used to form the special characters. After much experimentation and some pain, I finally succeeded, and now I'd like to share the result with you. When Accu-Print is controlling the printing, the program shown in Figure 1 will print on paper exactly as it's shown in the figure. As you can see, all Atari characters can be printed.

Accu-Print System Description

In its usual configuration, printing on the Atari computer is controlled by a routine within the Operating System ROM called the printer driver. Each time a character is to be printed, either by a cartridge (such as BASIC or the Assembler-Editor) or by an application program (such as the Atari Program Text Editor, APX Forms, or a user-written one), the printer driver is executed and sends the ATASCII code for the character to the printer. The printer then responds to the code according to its internal character set, *not the Atari's*. Under the Accu-Print system, a new printer driver replaces the one contained in the OS ROM. Once loaded, the new printer driver controls all subsequent printing by any program. The new printer driver is contained in an AUTORUN.SYS file and is located automatically and attached to the Operating System during system startup. To use the system, all you have to do is make sure AUTORUN.SYS file is on your boot disk and then start and use the computer system normally. No differences will be noticeable until Atari graphics or inverse characters are printed. Then, the special characters will simply be reproduced on the printer, although the printing will slow down somewhat due to the use of the printer's graphic mode.

There are basically two parts to the Accu-Print system. The first is the AUTORUN.SYS file which contains the new printer driver. The second is the Customizer program which is a BASIC program that creates the AUTORUN.SYS file for you. It allows you to select typestyle and control character options, and it permits you a great deal of flexibility in using Accu-Print and your printer. Customization of the AUTORUN.SYS file will be explained in a later section.

Getting Started

Listing 1 shows the Customizer program. When executed, it will create the AUTORUN.SYS file you need to do the accurate printing described above. For now, let's just try it to see how it works. Type the program into your computer exactly as shown in the listing. Don't forget to save it on your disk drive! You might call it CUST.BAS. When you've got it right, put a disk in your drive that doesn't have an AUTORUN.SYS file on it and run the Customizer program. Type 1 followed by a RETURN when you see the typestyle menu. Type N followed by a RETURN in response to the control character selection question. Finally, type D:AUTORUN.SYS followed by a RETURN in response to the file name question. The disk will now chug away, and in a few moments you'll see a question regarding creation of another file on the screen. Type N followed by a RETURN and the program will end. The Accu-Print printer driver is now embedded in the AUTORUN.SYS on your disk and is ready for your use.

Now, turn your computer off and then on again. When the disk stops spinning, the new printer driver has become an integral part of your computer's operating system. To try it out, just type in the listing shown in Figure 1 and list it to your printer. If all is well, your printer will print out an exact copy of Figure 1. Don't be concerned that the printer slows down when it encounters the special characters. It happens because the printer automatically does a carriage return when it goes into dot graphics mode, which is used to print the graphics and inverse characters.

With the system as it is now, you can print all of your program listings that contain graphics characters accurately on your printer. Just be sure to start your system using the disk containing the Accu-Print AUTORUN.SYS file. But since Accu-Print can do even more, let's get a little deeper into the system.

Figure 1

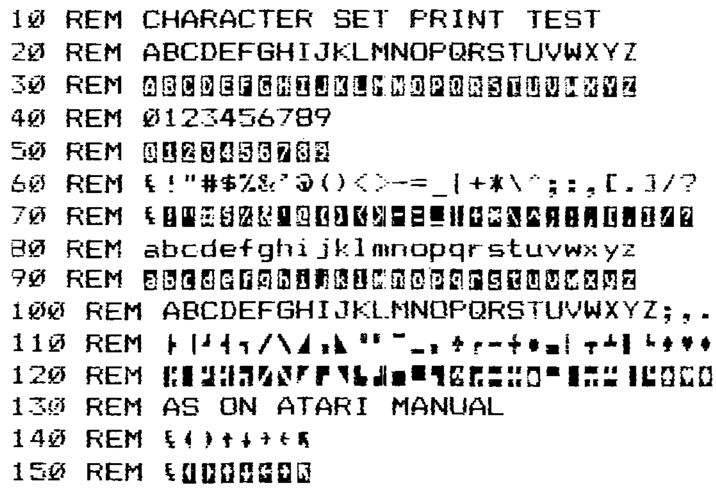


Figure 2

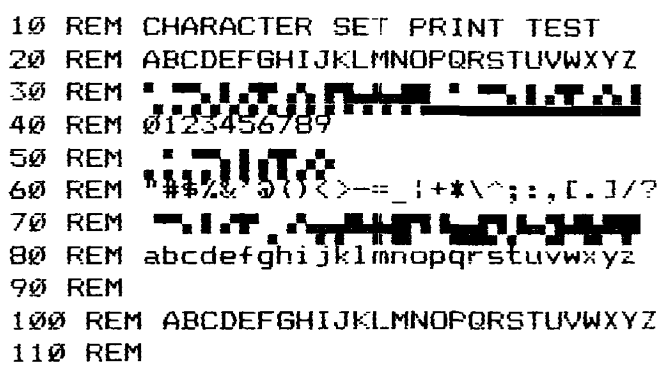


Figure 3

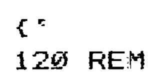
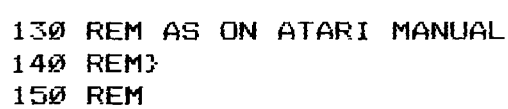


Figure 4



Listing 1

```
                   ; ACCU-PRINT
                   ;
                   ; THIS ROUTINE RUNS AS AN AUTO RUN ROUTINE
                   ; IT CAUSES THE OPERATING SYSTEM TO USE THE
                   ; NEW DEVICE WRITE PORTION OF THE PRINTER HANDLER.
                   ;
1CFC                        ORG $1CFC
                   ;
001D               ZINIT    EQU $1D
001E               ZCHAR    EQU $1E
001F               ZSAVE    EQU $1F
                   ;
02F4               CBASE    EQU $02F4
1D17               NPTAB    EQU $1D17
                   ;
1CFC AD 1F D0      BEGIN    LDA $D01F        ; CHECK FOR OPTION KEY
1CFF 29 04                  AND #$04
1D01 F0 13                  BEQ FINIS
                   ; IF = DON'T USE NEW PRINTER ROUTINE
1D03 A9 17                  LDA #NPTAB       ; REVECTOR PRINTER
1D05 8D 1B 03               STA $031B
1D08 A9 1D                  LDA /NPTAB
1D0A 8D 1C 03               STA $031C
1D0D EE E8 02               INC $2E8
1D10 EE E8 02               INC $2E8
1D13 EE E8 02               INC $2E8
1D16 60            FINIS    RTS              ; RETURN
                   ;
                   ; SUBROUTINE VECTORS
1D17 9E EE                  ADR $EE9E        ; OPEN
1D19 DB EE                  ADR $EEDB        ; CLOSE
1D1B 9D EE                  ADR $EE9D        ; READ
1D1D B7 1D                  ADR PWRIT-1      ; NEW WRITE
1D1F 80 EE                  ADR $EE80        ; STATUS
1D21 9D EE                  ADR $EE9D        ; SPECIAL
1D23 4C                     BYT $4C          ; JUMP VECTOR TO
1D24 78 EE                  ADR $EE78        ; DEVICE INIT ROUTINE
                   ;
                   ; NEW DEVICE WRITE ROUTINE
                   ;
1D26 00            DW       BYT 0            ; DOUBLE WIDE
1D27 00            CHCNT    BYT 0            ; LINE CHAR COUNT
1D28 01            CHINC    BYT 1            ; INCREMENT FOR GRAPHICS
1D29 51            LEN      BYT 81           ; LINE OVERFLOW LENGTH
1D2A 00            FT       BYT 0            ; FIRST TIME INDICATOR
1D2B 00            IS       BYT 0            ; PRINTER INIT STRING
1D2C 00 00 00               BYT 0,0,0,0,0,0,0,0
                   ;
                   ; NOTE: DUE TO SPACE CONSTRAINTS WE DO NOT LIST OUT
                   ; EVERY BYTE SEPARATELY, INSTEAD WE COMBINE THEM,
                   ; WE USE THIS CONVENTION PARTICULARLY WHEN LISTING TEXT
                   ;
1D34 00            AS       BYT 0            ; SPECIAL CONTROL CHAR ALLOW
1D35 00 00 00               DBY 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1D53 00 00 00               DBY 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1D71 00 00 00               DBY 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1D8F 00 00 00               DBY 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1DAD 00 00 00               DBY 0,0,0,0
1DB5 00            GR       BYT 0            ; GRAPHIC INDICATOR
1DB6 00            COLCT    BYT 0            ; CURRENT COLUMN OF CHAR
1DB7 00            FRET     BYT 0            ; FOUND 155 (CR) INDICATOR
                   ;
```

The OPTION Key

When the AUTORUN.SYS file containing the new printer driver is on one of your boot disks, it will automatically control the printing whenever you start your system using that disk. However, there may be times when you don't want the Accu-Print system to handle the printing. Instead of using another disk that doesn't contain Accu-Print's AUTORUN.SYS file, simply press the OPTION key and turn your computer on. Keep the OPTION key depressed until the disk drive stops and the startup operation is complete. Now printing will be controlled by the normal Operating System printer driver. To use Accu-Print again, just turn your computer off and restart your system without depressing the OPTION key.

Compatibility With Other Programs

The Accu-Print printer driver uses 768 bytes of memory, which is subtracted from the amount of memory available for application programs. The AUTORUN.SYS file, upon booting, modifies the LOMEM pointer so that application programs may be loaded and run in conjunction with the printer driver without overwriting it.

Since Accu-Print is designed to coexist and run with other programs, it is compatible with language cartridges such as BASIC, PILOT and Assembler-Editor. It is also compatible with standalone programs such as the Atari Program Text Editor and APX Forms. It is not compatible with any program that has its own AUTORUN.SYS file or that directly calls the Operating System's printer driver routine. In short, Accu-Print is compatible with any program or cartridge that uses or supports normal Atari Operating System printing conventions.

Control Characters

The Epson (and Epson-compatible) printers, without software support like Accu-Print, will not normally print Atari graphics or inverse characters. The reason is that while the Atari computer will display graphics symbols in response to certain numeric values, the Epson printers will consider those same values as control codes and respond accordingly. For example, the BASIC statement PRINT CHR$(12) will cause a graphics symbol to appear on the screen. However, the statement

LPRINT CHR$(12) will cause the Epson printers to form feed.

When the Accu-Print printer driver as created above is controlling the printing, the only control character the Epson printers will respond to is the carriage return. All other control characters will cause the printers to print the same graphics characters that would appear on the screen. For program listings and other uses, this is precisely what you want. However, there are other applications for which you would want your Epson printer to respond normally to certain control characters while responding to others by printing the Atari graphics or inverse symbols. For example, the Assembler-Editor cartridge generates form feed control characters when printing assembly listings. But you might want to put inverse characters in comment lines for emphasis. Therefore, you'd like to use the Accu-Print printer driver, but not have it treat form feeds as Atari graphics symbols. The Customizer program allows you to customize the Accu-Print printer driver for this type of use.

The Customizer Program

This program (Listing 1) creates an Accu-Print printer driver that has been customized for your use. As an option, you may select among any of 24 typestyles allowed by the Epson printers with Graftrax. All text except graphics or inverse characters will be printed in the typestyle you select. Note that you may not change typestyles while you are printing with the Accu-Print driver unless you select a control code option.

A second option is the specification of allowed control characters. You may specify up to 128 control characters which your Epson printer will respond to normally. That is, the Atari graphics or inverse characters corresponding to these symbols will not be printed in response to these control codes. Instead, the printer will respond as specified in its instruction manual.

Running the Customizer program is simple. With the BASIC language cartridge inserted, just load and run CUST.BAS. After initialization is completed, you will see a menu of typestyle choices. Just type the number corresponding to the typestyle you want followed by the RETURN key. You will then be asked if you want to allow control characters. If you do, type Y and RETURN in response to the question:

```
1DB8 85 1F      PWRIT   STA ZSAVE      ; STORE ATASCII CHAR
1DBA 20 1A EF           JSR $EF1A
1DBD A0 00              LDY #0         ; INIT COLUMN COUNT
1DBF 8C B5 1D           STY GR         ; AND INDICATORS
1DC2 8C B6 1D           STY COLCT
1DC5 8C B7 1D           STY FRET
1DC8 CC 2A 1D           CPY FT         ; FIRST TIME THRU ?
1DCB D0 13              BNE GOON       ; BRANCH IF NOT
1DCD EE 2A 1D           INC FT         ; BYPASS AFTER THIS
1DD0 B9 2B 1D   LOP     LDA IS,Y       ; CHECK FOR MORE
1DD3 C9 00              CMP #0         ; PRINTER INIT CHARS
1DD5 F0 09              BEQ GOON       ; BRANCH IF NO MORE
1DD7 A6 1D              LDX ZINIT      ; INIT CHAR IN PRINTER
1DD9 20 CF 1E           JSR STCHR
1DDC C8                 INY            ; POINT TO NEXT INIT
1DDD 4C D0 1D           JMP LOP        ; CHAR AND GET IT
1DE0 A5 1F      GOON    LDA ZSAVE
1DE2 20 0A 1F           JSR CHRCK      ; CHECK FOR NON-COMPATIBLE
1DE5 C0 01              CPY #1         ; CHARACTER
1DE7 F0 03              BEQ INCMP      ; BRANCH IF INCOMPATIBLE
1DE9 4C BC 1E           JMP CMPAT      ; COMPATIBLE
1DEC EE B5 1D   INCMP   INC GR         ; SET GRAPHIC FLAG
1DEF 18                 CLC
1DF0 AD 28 1D           LDA CHINC      ; INCREMENT LINE
1DF3 6D 27 1D           ADC CHCNT      ; CHAR COUNT
1DF6 8D 27 1D           STA CHCNT
1DF9 CD 29 1D           CMP LEN        ; CHECK FOR LINE OVERFLOW
1DFC D0 0D              BNE CTRL       ; BRANCH IF NO OVERFLOW
1DFE A9 9B              LDA #155       ; ELSE, SEND CARRIAGE RETURN
1E00 A6 1D              LDX ZINIT
1E02 20 CF 1E           JSR STCHR
1E05 AD 28 1D           LDA CHINC      ; CLEAR CHAR COUNT
1E08 8D 27 1D           STA CHCNT
1E0B A9 1B      CTRL    LDA #$1B       ; PUT GRAPHICS CONTROL
1E0D A6 1D              LDX ZINIT      ; CHARS INTO PRINTER BUFFER
1E0F 20 CF 1E           JSR STCHR      ; STORE CHAR
1E12 A9 4C              LDA #76
1E14 AC 26 1D           LDY DW         ; CHECK FOR DOUBLE WIDE
1E17 C0 00              CPY #0
1E19 F0 02              BEQ LD76       ; BRANCH IF NOT
1E1B A9 4B              LDA #75
1E1D A6 1D      LD76    LDX ZINIT
1E1F 20 CF 1E           JSR STCHR
1E22 A9 08              LDA #8
1E24 A6 1D              LDX ZINIT
1E26 20 CF 1E           JSR STCHR
1E29 A9 00              LDA #0
1E2B A6 1D              LDX ZINIT
1E2D 20 CF 1E           JSR STCHR
1E30 AC B6 1D   BACK    LDY COLCT      ; COLUMN COUNT IN Y
1E33 A5 1F              LDA ZSAVE      ; ATASCII CODE IN A
1E35 20 48 1F           JSR GETCL      ; GET COLUMN VALUD
1E38 8A                 TXA            ; PUT INTO A
1E39 C9 9B              CMP #155       ; 155 IS ATASCII EOL
1E3B D0 05              BNE PUTC
1E3D EE B7 1D           INC FRET       ; SET FOUND 155 FLAG
1E40 A9 90              LDA #144       ; SEND A 144 INSTEAD
1E42 A6 1D      PUTC    LDX ZINIT      ; INDEX INTO PRINTER BUFFER
1E44 EE B6 1D           INC COLCT      ; SET CLOUMN COUNT
1E47 20 CF 1E           JSR STCHR      ; PUT COLUMN VALUE IN PB
1E4A A9 08              LDA #8
1E4C CD B6 1D           CMP COLCT
1ED9 C9 9B              CMP #$9B
```

```
1E4F D0 DF            BNE BACK      ; BRANCH IF NOT LAST COLUMN
1E51 A9 00            LDA #0        ; IF FRET SET
1E53 CD B7 1D         CMP FRET      ; NEED TO REPEAT CHAR
1E56 F0 61            BEQ ORD
1E58 8D B6 1D         STA COLCT     ; ZERO COLCT
1E5B AC 28 1D         LDY CHINC
1E5E A9 08     BS1    LDA #8        ; PUT BACKSPACE IN BUFFER
1E60 A6 1D            LDX ZINIT
1E62 20 CF 1E         JSR STCHR
1E65 AC 26 1D         LDY DW        ; CHECK FOR DOUBLE WIDE
1E68 C0 00            CPY #0
1E6A F0 07            BEQ BS2       ; BRANCH IF NOT
1E6C A9 08            LDA #8        ; PUT IN ANOTHER
1E6E A6 1D            LDX ZINIT     ; BACKSPACE
1E70 20 CF 1E         JSR STCHR
1E73 A9 1B     BS2    LDA #$1B      ; PUT GRAPHICS CONTROL
1E75 A6 1D            LDX ZINIT     ; CHAR IN BUFFER
1E77 20 CF 1E         JSR STCHR
1E7A A9 4C            LDA #76
1E7C AC 26 1D         LDY DW
1E7F C0 00            CPY #0
1E81 F0 02            BEQ BS3       ; BRANCH IF NOT DOUBLE WIDE
1E83 A9 4B            LDA #75
1E85 A6 1D     BS3    LDX ZINIT
1E87 20 CF 1E         JSR STCHR
1E8A A9 08            LDA #8
1E8C A6 1D            LDX ZINIT
1E8E 20 CF 1E         JSR STCHR
1E91 A9 00            LDA #0
1E93 A6 1D            LDX ZINIT
1E95 20 CF 1E         JSR STCHR
1E98 AC B6 1D  BK1    LDY COLCT     ; COLUMN COUNT IN Y
1E9B A5 1F            LDA ZSAVE     ; ATASCII IN A
1E9D 20 48 1F         JSR GETCL     ; COLUMN VALUE
1EA0 8A               TXA           ; INTO A
1EA1 A2 00            LDX #0
1EA3 C9 9B            CMP #155      ; TEST FOR 155
1EA5 D0 02            BNE OK        ; BRANCH IF NOT 155
1EA7 A2 0B            LDX #11       ; REPLACE 155 WITH 11
1EA9 8A        OK     TXA           ; PUT CORRECT VALUE IN A
1EAA A6 1D            LDX ZINIT     ; INDEX INTO PB IN X
1EAC EE B6 1D         INC COLCT     ; SET COLUMN COUNT
1EAF 20 CF 1E         JSR STCHR     ; PUT COLUMN VALUE IN PB
1EB2 A9 08            LDA #8
1EB4 CD B6 1D         CMP COLCT
1EB7 D0 DF            BNE BK1       ; BRANCH IF NOT LAST COLUMN
1EB9 A0 01     ORD    LDY #$01
1EBB 60               RTS           ; RETURN
1EBC AD 27 1D  CMPAT  LDA CHCNT     ; INC. CHAR COUNT
1EBF 6D 28 1D         ADC CHINC
1EC2 8D 27 1D         STA CHCNT
1EC5 A5 1F            LDA ZSAVE
1EC7 A6 1D            LDX ZINIT
1EC9 20 CF 1E         JSR STCHR     ; STORE CHAR IN PB
1ECC 4C B9 1E         JMP ORD       ; AND EXIT
               ;
               ; STORE CHAR IN A INTO PB AT OFFSET X
               ; PRINT IF BUFFER FULL OR CR.
               ;
1ECF 9D C0 03  STCHR  STA $03C0,X   ; STORE CHAR
1ED2 E8               INX
1ED3 E4 1E            CPX ZCHAR
1ED5 F0 16            BEQ LAST      ; BRANCH IF LAST CHAR FOR PB
1ED7 86 1D            STX ZINIT     ; UPDATE POINTERS
```

ANY CONTROL CHARACTERS TO ALLOW?

Then type the decimal value of the control character you wish your Epson printer to respond to normally followed by the RETURN key. You will then be asked:

ANY MORE?

Respond with a Y and RETURN to specify more control characters, and continue in this way until you have typed all the control characters you wish to allow. You can specify up to 128 control characters in this manner.

You will then be asked for a file name. This is the name of the file your customized printer driver will be written to. You may give any legal file name and must give the complete specifier, including the disk drive. For example, this could be D1:SPECIAL.OBJ. Follow the file specification with the RETURN key. The file will then be written to the disk. Next you will be asked if you want to create another printer driver file. Type Y and RETURN if you want to create another customized printer driver, or N and RETURN if you want to exit the program.

To use your new printer driver, just copy the file you created to the AUTORUN.SYS file and reboot your system with the disk containing it. You should also turn your printer off and then on again before you use a new printer driver so it will be cleared of any previous settings.

You will probably want to create several Accu-Print printer drivers for different uses. For example, you might have one using normal Pica type and no control characters allowed for BASIC program listings, another using Pica type and allowing form feeds for assembly listings using the Assembler-Editor cartridge, and perhaps another using Emphasized Pica type and allowing several control characters for word processing applications.

Additional Details

Pressing the RESET key will make the Accu-Print printer driver inactive. You will have to restart your Atari computer system to use Accu-Print again.

Accu-Print uses the character definitions stored inside your Atari computer to generate the graphics and inverse characters on your printer. In fact, it uses the CHBAS Operating System vector to find the character set definitions in memory. Thus, if you use the Accu-Print printer driver with an application program that uses a redefined character set, the redefined characters will be printed on your printer. This is useful for many special applications, and can be the basis for special graphics character screen dump programs.

How Accu-Print Works

Listing 2 is an assembly language listing of the Accu-Print printer driver. I've included it for those of you who might like to understand how the printer driver works. Additionally, you might wish to modify it for your own special purposes. I've tried to liberally comment the listing to make it a little easier to understand.

The basic idea behind Accu-Print is to replace the normal Operating System printer driver with one of my own design. This is made possible by two features of the Atari system. The first is the capability to load and execute a program stored in an AUTORUN.SYS file at system startup after the system is initialized, but before the user is given control of the system. The second is the fact that the Operating System uses RAM to store pointers (or vectors) to input/output control routines. Combining these features, the system allows us to execute a program (continued in AUTORUN.sys) during system startup that changes the print pointer from the standard printer driver to our own. Labels BEGIN through FINIS show these operations. At system startup, the entire AUTORUN.SYS file is loaded into memory and execution begin at BEGIN. If the OPTION key is pressed, the program simply exits and nothing happens. Otherwise, the address of our new printer device table (located at NPTAB) is stored in locations $31B and $31C, which contain the pointer to the Operating System's standard printer device table. Next, the LOMEM pointer is incremented by 768 to make sure the following code isn't overwritten by an application. The program then exits and the user is given control of the system. Only a few instructions are executed at system startup, but the effect is great! All printing will now be vectored through our new printer driver. The data for the driver starts at DW and the executable code starts at PWRIT.

Chapter 8 of the De Re Atari gives more information about Operating System vectors and device tables if you're interested in more detail about

```
1EDB F0 01              BEQ CR          ; BRANCH IF CHAR IS EOL
1EDD 60        NEXT     RTS             ; RETURN
1EDE A9 20     CR       LDA #$20
1EE0 A0 00              LDY #0
1EE2 8C 27 1D           STY CHCNT       ; CLEAR CHAR COUNT FOR NEW LIN
1EE5 9D C0 03  NXT      STA $03C0,X     ; PAD BUFFER WITH
1EE8 E8                 INX             ; BLANKS
1EE9 E4 1E              CPX ZCHAR
1EEB D0 F8              BNE NXT
1EED A9 00     LAST     LDA #$00        ; SEND BUFFER TO PRINTER
1EEF 85 1D              STA ZINIT
1EF1 AE 7F EE           LDX $EE7F
1EF4 AC 80 EE           LDY $EE80
1EF7 20 E6 EE           JSR $EEE6
1EFA 20 59 E4           JSR $E459
1EFD AD 26 1D           LDA DW          ; CHECK DOUBLE WIDE
1F00 C9 00              CMP #0
1F02 F0 05              BEQ OUT         ; BRANCH IF NOT
1F04 A9 00              LDA #0          ; FORCT
1F06 8D 2A 1D           STA FT          ; INITIALIZATION
1F09 60        OUT      RTS             ; RETURN
               ;
               ; CHECK CHAR FOR COMPATIBILITY WITH
               ; PRINTER CHAR SET. RETURN Y=0 IF COMPATIBLE,
               ; Y=1 IF NOT. ATASCII VALUES INCOMPATIBLE:
               ; 0-31,96,123-154,156-255
               ;
1F0A 38        CHRCK    SEC
1F0B C9 9C              CMP #156
1F0D B0 27              BCS SONE        ; >155 - NEED GRAPHICS
1F0F C9 9B              CMP #155
1F11 F0 20              BEQ SZERO       ; = 155 - DON'T NEED GRAPHICS
1F13 A0 00              LDY #0
1F15 BE 34 1D  ANXT     LDX AS,Y        ; CHECK FOR SPECIAL
1F18 E0 00              CPX #0          ; ALLOW CHARACTERS
1F1A F0 09              BEQ NMOR        ; BRANCH IF NO MORE
1F1C D9 34 1D           CMP AS,Y        ; CHECK CHAR
1F1F F0 12              BEQ SZERO       ; ALLOW IT IF EQUAL
1F21 C8                 INY             ; POINT TO NEXT ALLOW CHAR
1F22 4C 15 1F           JMP ANXT        ; AND CHECK I
1F25 38        NMOR     SEC
1F26 C9 7B              CMP #123
1F28 B0 0C              BCS SONE        ; >122 - NEED GRAPHICS
1F2A 38                 SEC
1F2B C9 20              CMP #32
1F2D 90 07              BCC SONE        ; <32 - NEED GRAPHICS
1F2F C9 60              CMP #96
1F31 F0 03              BEQ SONE        ; = 96 - NEED GRAPHICS
1F33 A0 00     SZERO    LDY #0          ; DON'T NEED GRAPHICS
1F35 60                 RTS
1F36 A0 01     SONE     LDY #1          ; NEED GRAPHICS
1F38 60                 RTS
               ;
               ; COMPUTE COLUMN VALUE TO CONSTRUCT CHAR
               ; A = ATASCII VALUE OF CHAR
               ; Y = COLUMN VALUE TO COMPUTE
               ; X = COLUMN VALUE ON EXIT
               ;
1F39 00        ANS      BYT 0           ; COMPUTER COLUMN VALUE
1F3A 00        ATVAL    BYT 0           ; ATASCII CHAR VALUE
1F3B 00        CVAL     BYT 0           ; INTERNAL CHAR VALUE
1F3C 00        CLNUM    BYT 0           ; COLUMN NUMBER
1F3D 00        INFLG    BYT 0           ; INVERSE CHAR FLAG
```

```
1F3E 00 00      CHBAS   DBY 0           ; CHAR SET BASE ADDRESS
1F40 80 40 20   MASKS   BYT 128,64,32,16,8,4,2,1 ; MASKS FOR COLUMNS 0 TO
7
                ;
                ; SEE 'NOTE' IN FIRST PAGE OF LISTING
                ;
1F48 8D 3A 1F   GETCL   STA ATVAL       ; SAVE ATASCII VALUE
1F4B 8C 3C 1F           STY CLNUM       ; SAVE COLUMN NUMBER
1F4E A0 00              LDY #0          ; INITIALIZE
1F50 8C 39 1F           STY ANS
1F53 38                 SEC
1F54 C9 80              CMP #128
1F56 90 0B              BCC NCHAR       ; BRANCH IF NOT INVERSE
1F58 AD 3A 1F           LDA ATVAL
1F5B 38                 SEC
1F5C E9 80              SBC #128        ; CONVERT TO NON-INVERSE
1F5E 8D 3A 1F           STA ATVAL
1F61 A0 01              LDY #1
1F63 8C 3D 1F   NCHAR   STY INFLG       ; SET INVERSE FLAG
1F66 AC F4 02           LDY CBASE       ; GET CHAR SET BASE
1F69 8D 3F 1F           STA CHBAS+1     ; AND SAVE IT
1F6C AD 3A 1F           LDA ATVAL       ; CONVERT ATVAL TO
1F6F 38                 SEC
1F70 C9 60              CMP #$60        ; INTERNAL CODE
1F72 90 04              BCC L60         ; BRANCH IF <$60
1F74 4C 89 1F           JMP CSTOR       ; ELSE, CONTINUE
1F77 38                 SEC
1F78 C9 40      L60     CMP #$40
1F7A 90 06              BCC L40         ; BRANCH IF <$40
1F7C 38         G20     SEC
1F7D E9 20              SBC #$20
1F7F 4C 89 1F           JMP CSTOR
1F82 38         L40     SEC
1F83 C9 20              CMP #$20
1F85 B0 F5              BCS G20         ; BRANCH IF >=$20
1F87 69 40              ADC #$40
1F89 8D 3B 1F   CSTOR   STA CVAL        ; STORE INTERNAL VALUE
1F8C AA                 TAX             ; COMPUTE ADDRESS OF CHAR
1F8D A9 00              LDA #0          ; DEFINITION.
1F8F E0 00      LOOP    CPX #0
1F91 F0 0E              BEQ CMPCL       ; NO MORE ADJUSTMENT NECESSARY
1F93 18                 CLC
1F94 69 08              ADC #8
1F96 CA                 DEX             ; DECR. INTERNAL VALUE
1F97 90 F6              BCC LOOP        ; SEE IF OFFSET = 256
1F99 EE 3F 1F           INC CHBAS+1     ; BUMP HIGH
1F9C A9 00              LDA #0
1F9E 4C 8F 1F           JMP LOOP
1FA1 8D 3E 1F   CMPCL   STA CHBAS       ; STORE LOW BYTE
1FA4 8D B6 1F           STA MOD1+1
1FA7 AD 3F 1F           LDA CHBAS+1
1FAA 8D B7 1F           STA MOD1+2
1FAD A0 00              LDY #0          ; CONTROLS LOOP
1FAF AE 3C 1F           LDX CLNUM       ; INDEX TO MASKS
1FB2 BD 40 1F   LOOP1   LDA MASKS,X     ; GET COLUMN MASK
1FB5 39 3E 1F   MOD1    AND CHBAS,Y     ; MODIFIED - WILL POINT TO
1FB8 C9 00              CMP #0          ; CHAR. DEFINITION
1FBA F0 0A              BEQ CHECK       ; BRANCH IF COLUMN BIT NOT SET
1FBC AD 39 1F           LDA ANS         ; ELSE, UPDATE COLUMN VALUE
1FBF 18                 CLC
1FC0 79 40 1F           ADC MASKS,Y     ; ADD VALUE TO AND
1FC3 8D 39 1F           STA ANS
1FC6 C8         CHECK   INY
1FC7 C0 08              CPY #8
```

that area. To set the stage for understanding the printer driver code itself, let's notice that when a character is to be printed, its ATASCII code will be placed in the A-register and the code beginning at PWRIT will be executed.

The driver first checks to see if this is the very first time the printer driver is being executed. If it is, then the characters contained in the data string starting at IS will be sent to the printer to initialize it. The particular character string stored here is a function of the typestyle selected when you ran the Customizer program. The driver then checks to see if the character to be printed is a graphics or inverse character. The subroutine at CHRCK is used for this. If it is not a special character, then subsequent code is bypassed and the driver operates exactly like the standard one.

If a special character is to be printed, then some special processing takes place. Each special character is printed in bit graphics mode. This means that data values corresponding to the individual 8-dot columns of the printed characters have to be sent to the printer in addition to control characters putting the printer into and out of graphics mode. This accounts for the slowdown while printing these characters. For each special character printed, 12 data characters have to be sent. In addition, special handling has to be given if a column data value happens to be 155. The Atari system will recognize this value as a carriage return and send a line feed character after it. Since we really want this value to be printed as a single 8-dot column, the automatic insertion of additional data is unacceptable. So if this value occurs (as it does with an inverse A) the driver breaks it apart, prints part of the column, backspaces the printer, and prints the second part. At any rate, the code between INCMP and CMPAT is devoted to sending data to the printer that causes it to print the 8-dot columns that form the graphics or inverse character being printed. After this data is sent, the printer is taken out of graphics mode and the printer driver is exited.

Two subroutines worthy of note are CHRCK and GETCL CHRCK determines when a character needs special handling. It does this by checking the character's ATASCII code with the codes of the graphics and inverse characters. Also, it checks the character's code against the list of legal control codes (if any) you specified while running the Customizer program. The character string AS contains those control codes. So it is this routine which allows for passing certain control characters intact to the printer.

GETCL is the real workhorse of the printer driver. It accesses the internal character definitions and computes the data values to send to the printer so it can reconstruct the characters precisely as the Atari defines them. The routine is executed 8 times for each special character, once for each 8-dot column. The algorithm used is interesting, since it has to translate between the row-by-row internal character set definitions and the column-by-column data required by the printer. You can also see why Accu-Print works with custom character sets. It uses the standard character set vector to find the character definitions. Custom character sets use this vector too! My technique is probably not the most sophisticated possible. An interesting exercise would be to make it shorter and more efficient. I have a feeling that one of you whizzes out there can write this subroutine using one quarter of the code I did. Any takers?

Well, I hope this explanation of how the code works will help you understand some of the subtleties of the Atari Operating System and of assembly language. I highly recommend De Re Atari and the Technical Reference Notes for more in-depth treatments of the techniques used.

A Concluding Note

I've found the Accu-Print system to be very useful in my work. I can now feel free to use graphics strings in my programs at will, particularly to represent assembly language routines where it saves me a lot of typing and leads to faster execution times for initialization. I don't get gibberish on my printer anymore, and I find that most programs work with Accu-Print easily. I hope you find the system helpful to you as well.

```
1FC9 D0 E7              BNE LOOP1       ; BRANCH IF NOT DONE
1FCB AE 3D 1F           LDX INFLG       ; HANDLE INVERSE
1FCE E0 01              CPX #1
1FD0 D0 08              BNE FIN         ; BRANCH IF NOT INVERSE
1FD2 A9 FF              LDA #255        ; ELSE, FLIP BITS
1FD4 4D 39 1F           EOR ANS
1FD7 8D 39 1F           STA ANS
1FDA AD 3A 1F   FIN     LDA ATVAL       ; LOAD REGS FOR EXIT
1FDD AC 3C 1F           LDY CLNUM
1FE0 AE 39 1F           LDX ANS
1FE3 60                 RTS
                ;
1FE4                    END
```

Listing 2

```
1 REM ACCU-PRINT CUSTOMIZER
4 REM
10 GRAPHICS 0:POSITION 15,8:? "ACCU-PRINT"
20 POSITION 15,10:? "CUSTOMIZER"
40 POSITION 5,15:? "(REVERSE ON)PLEASE WAIT FOR
INITIALIZATION(REVERSE OFF)"
50 REM PUT ACCU-PRINT IN STRING
60 DIM R$(744):RESTORE 900
70 FOR I=1 TO 744:READ A:R$(LEN(R$)+1)=CHR$(A):NEXT I
80 GRAPHICS 0:POSITION 15,0:? "ACCU-PRINT"
90 POSITION 13,1:? "TYPESTYLE MENU"
100 POSITION 2,3:? " 1 PICA"
110 POSITION 2,4:? " 2 ITAL"
120 POSITION 2,5:? " 3 PICA EMPH"
130 POSITION 2,6:? " 4 ITAL EMPH"
140 POSITION 2,7:? " 5 PICA DS"
150 POSITION 2,8:? " 6 ITAL DS"
160 POSITION 2,9:? " 7 PICA EMPH DS"
170 POSITION 2,10:? " 8 ITAL EMPH DS"
180 POSITION 2,11:? " 9 COND PICA"
190 POSITION 2,12:? "10 COND ITAL"
200 POSITION 2,13:? "11 COND PICA DS"
210 POSITION 2,14:? "12 COND ITAL DS"
220 POSITION 20,3:? "13 COND-EXP PICA"
230 POSITION 20,4:? "14 COND-EXP ITAL"
240 POSITION 20,5:? "15 COND-EXP PICA DS";
250 POSITION 20,6:? "16 COND-EXP ITAL DS";
260 POSITION 20,7:? "17 EXP PICA"
270 POSITION 20,8:? "18 EXP ITAL"
280 POSITION 20,9:? "19 EXP PICA EMPH"
290 POSITION 20,10:? "20 EXP ITAL EMPH"
300 POSITION 20,11:? "21 EXP PICA DS"
310 POSITION 20,12:? "22 EXP ITAL DS"
320 POSITION 20,13:? "23 EXP PICA EMPH DS";
330 POSITION 20,14:? "24 EXP ITAL EMPH DS";
340 POSITION 2,16:? "YOUR CHOICE";
350 INPUT CH
360 ON CH GOTO 400,405,410,415,420,425,430,435,440,445,
450,455,460,465,470,475,480,485,490,495,500,505,510,515
370 PRINT CHR$(253);:GOTO 340
400 RESTORE 401:GOTO 600
401 DATA 255
405 RESTORE 406:GOTO 600
406 DATA 27,52,255
410 RESTORE 411:GOTO 600
411 DATA 27,69,255
415 RESTORE 416:GOTO 600
416 DATA 27,52,27,69,255
420 RESTORE 421:GOTO 600
```

```
421 DATA 27,71,255
425 RESTORE 426:GOTO 600
426 DATA 27,52,27,71,255
430 RESTORE 431:GOTO 600
431 DATA 27,69,27,71,255
435 RESTORE 436:GOTO 600
436 DATA 27,52,27,69,27,71,255
440 RESTORE 441:GOTO 540
441 DATA 27,80,255
445 RESTORE 446:GOTO 540
446 DATA 27,80,27,52,255
450 RESTORE 451:GOTO 540
451 DATA 27,80,27,71,255
455 RESTORE 456:GOTO 540
456 DATA 27,80,27,52,27,71,255
460 RESTORE 461:GOTO 560
461 DATA 15,14,255
465 RESTORE 466:GOTO 560
466 DATA 15,14,27,52,255
470 RESTORE 471:GOTO 560
471 DATA 15,14,27,71,255
475 RESTORE 476:GOTO 560
476 DATA 15,14,27,52,27,71,255
480 RESTORE 481:GOTO 580
481 DATA 14,255
485 RESTORE 486:GOTO 580
486 DATA 14,27,52,255
490 RESTORE 491:GOTO 580
491 DATA 14,27,69,255
495 RESTORE 496:GOTO 580
496 DATA 14,27,52,27,69,255
500 RESTORE 501:GOTO 580
501 DATA 14,27,71,255
505 RESTORE 506:GOTO 580
506 DATA 14,27,52,27,71,255
510 RESTORE 511:GOTO 580
511 DATA 14,27,69,27,71,255
515 RESTORE 516:GOTO 580
516 DATA 14,27,52,27,69,27,71,255
540 R$(46,46)=CHR$(133):GOTO 600
560 R$(46,46)=CHR$(67):R$(43,43)=CHR$(1):GOTO 600
580 R$(46,46)=CHR$(41):R$(43,43)=CHR$(1)
600 I=48
610 READ A:IF A=255 THEN GOTO 700
620 R$(I,I)=CHR$(A):I=I+1:GOTO 610
700 GRAPHICS 0:POSITION 15,0:? "ACCU-PRINT"
710 POSITION 7,1:? "CONTROL CHARACTER SELECTION":
    POSITION 2,3:I=57
720 PRINT "ANY CONTROL CHARACTERS TO ALLOW";
730 DIM A$(1):INPUT A$:IF A$(1,1)<>"Y" THEN GOTO 800
740 PRINT "CONTROL CHARACTER TO ALLOW: ";
750 INPUT A:R$(I,I)=CHR$(A):I=I+1:IF I>185 THEN GOTO 800
760 PRINT "ANY MORE";
770 INPUT A$:IF A$(1,1)="Y" THEN GOTO 740
800 GRAPHICS 0:POSITION 15,0:? "ACCU-PRINT"
810 POSITION 11,1:? "DISK FILE CREATION"
820 POSITION 2,3:? "FILE NAME: ";
830 DIM F$(15):INPUT F$
840 OPEN #1,8,0,F$
850 POSITION 2,5:
    ? "{REVERSE ON}NOW WRITING FILE{REVERSE OFF}"
855 PUT #1,255:PUT #1,255:PUT #1,252:PUT #1,28:
    PUT #1,227:PUT #1,31
860 FOR I=1 TO 744:PUT #1,ASC(R$(I,I)):NEXT I
865 PUT #1,224:PUT #1,2:PUT #1,225:PUT #1,2:
    PUT #1,252:PUT #1,28
870 CLOSE #1
875 POSITION 2,7:? "CREATE ANOTHER FILE";
880 INPUT A$:IF A$(1,1)="Y" THEN CLR :GOTO 10
890 GRAPHICS 0:CLR :END
900 DATA 173,31,208,41,4,240,19,169,23,141,27
901 DATA 3,169,29,141,28,3,238
902 DATA 232,2,238,232,2,238,232,2,96,158,238,219
903 DATA 238,157,238,183,29,128,238,157,238,76
904 DATA 120,238,0,0,1,81,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
905 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0
906 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
907 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
908 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
909 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
910 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
911 DATA 0,0,0,0,0,0,0,0,0,133,31,32,26,239,160
912 DATA 0,140,181,29,140,182,29,140,183,29,204,42
913 DATA 29,208,19,238,42,29,185,43,29,201,0
914 DATA 240,9,166,29,32,207,30,200,76,208,29,165
915 DATA 31,32,10,31,192,1,240,3,76,188,30,238,181
916 DATA 29,24,173,40,29,109,39,29,141,39,29,205
917 DATA 41,29,208,13,169,155,166,29,32,207,30
918 DATA 173,40,29,141,39,29,169,27,166,29,32
919 DATA 207,30,169,76,172,38,29,192,0,240,2,169,75
920 DATA 166,29,32,207,30,169,8,166,29,32,207,30,169
921 DATA 0,166,29,32,207,30,172,182,29,165,31
922 DATA 32,72,31,138,201,155,208,5,238,183,29
923 DATA 169,144,166,29,238,182,29,32,207,30,169
924 DATA 8,205,182,29,208,223,169,0,205,183,29
925 DATA 240,97,141,182,29,172,40,29,169,8,166,29,32
926 DATA 207,30,172,38,29,192,0,240,7,169,8,166
927 DATA 29,32,207,30,169,27,166,29,32,207,30,169
928 DATA 76,172,38,29,192,0,240,2,169,75,166,29,32
929 DATA 207,30,169,8,166,29,32,207,30,169,0
930 DATA 166,29,32,207,30,172,182,29,165,31,32
931 DATA 72,31,138,162,0,201,155,208,2,162,11,138,166
932 DATA 29,238,182,29,32,207,30,169,8,205,182,29,208
933 DATA 223,160,1,96,173,39,29,109,40,29
934 DATA 141,39,29,165,31,166,29,32,207,30,76,185,30
935 DATA 157,192,3,232,228,30,240,22,134,29
936 DATA 201,155,240,1,96,169,32,160,0,140,39,29,157
937 DATA 192,3,232,228,30,208,248,169,0,133,29
938 DATA 174,127,238,172,128,238,32,230,238,32,89,228
939 DATA 173,38,29,201,0,240,5,169,0,141,42
940 DATA 29,96,56,201,156,176,39,201,155,240,32,160,0
941 DATA 190,52,29,224,0,240,9,217,52,29,240
942 DATA 18,200,76,21,31,56,201,123,176,12,56,201,32
943 DATA 144,7,201,96,240,3,160,0,96,160,1
944 DATA 96,0,0,0,0,0,0,0,128,64,32,16,8,4,2,1,141,58
945 DATA 31,140,60,31,160,0,140,57,31,56,201
946 DATA 128,144,11,173,58,31,56,233,128,141,58,31,160
947 DATA 1,140,61,31,173,244,2,141,63,31
948 DATA 173,58,31,56,201,96,144,4,76,137,31,56,201,64
949 DATA 144,6,56,233,32,76,137,31,56,201,32
950 DATA 176,245,105,64,141,59,31,170,169,0,224,0,240
951 DATA 14,24,105,8,202,144,246,238,63,31
952 DATA 169,0,76,143,31,141,62,31,141,182,31,173,63,31
953 DATA 141,183,31,160,0,174,60,31,189,64
954 DATA 31,57,62,31,201,0,240,10,173,57,31,24,121,64,31
955 DATA 141,57,31,200,192,8,208,231,174
956 DATA 61,31,224,1,208,8,169,255,77,57,31,141,57,31
957 DATA 173,58,31,172,60,31,174,57,31,96
```

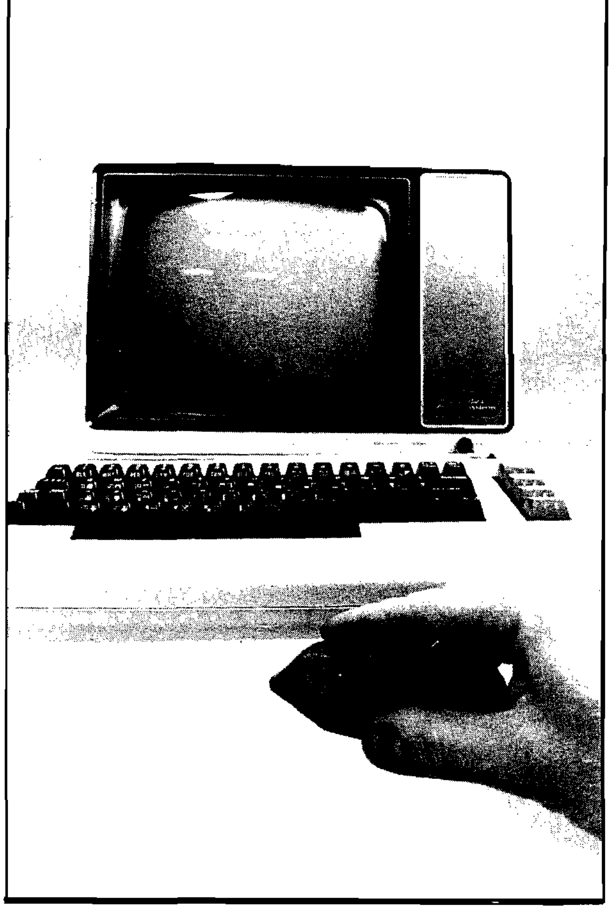


Figure 1. The mouse in use on the VIC-20.

A Low Cost Mouse for the VIC-20

by Robert L. Martin WB2KTG

As most readers of this magazine are aware, the "mouse" is a popular easy-to-use device for inputting data to computer or terminal. Many newly designed computers, such as Apple's Macintosh, are being built with mice as standard factory equipment.

Having recently purchased a VIC-20, and being unwilling to spend several thousand dollars to get a new mouse-equipped computer, I decided to build a mouse which could be used to upgrade my present system.

To begin this project I decided on the objectives of the design. First, my mouse should be a "hardware-only" design. I don't enjoy programming and, besides, it will be more of a challenge this way. Second, if I do upgrade my hardware at some time in the future, I don't want the mouse to be incompatible with whatever it is I buy. The mouse shall be usable with all computer systems in existence or planned. Third, the design should be simple enough that anyone could make a duplicate in one evening's time. And fourth, it should not be expensive. As you will soon see, these objectives limit the performance of the final product, but we do produce a mouse.

Not wishing to be inconvenienced by the care and feeding of a live mouse, I decided to start with the next best thing. A rubber mouse from the local pet store looked great. The mouse I bought cost less than two dollars. As an added bonus, my mouse was available in several colors.

A quick incision on the mouse's lower abdomen (no anesthetic necessary) with my trusty Swiss Army Knife and a control port was available for interconnecting cable insertion. I used a telephone extension cord with modular end connectors. The connector keeps the cable from pulling

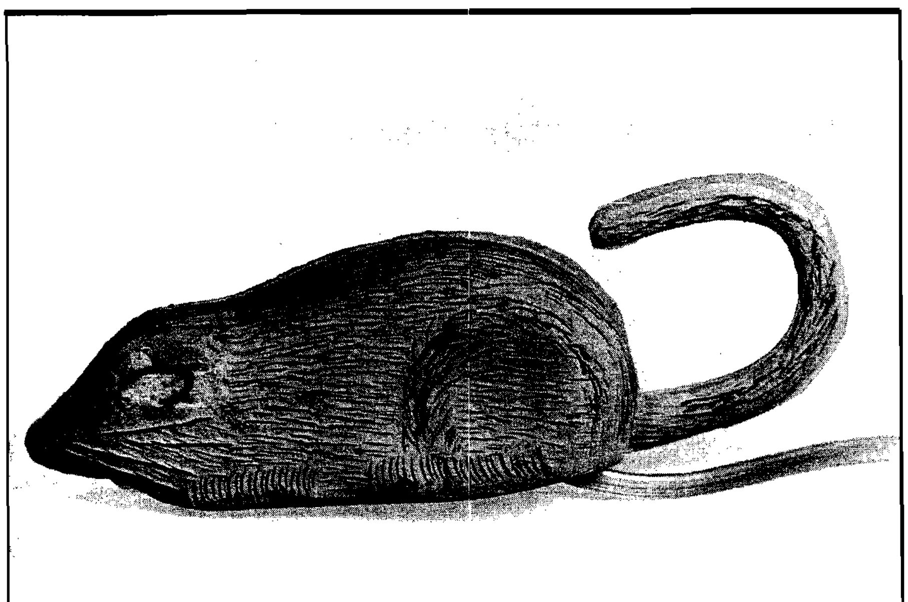


Figure 2. Detailed illustration of the mouse.

out of the control port. The other end of the cable is secured to the computer with a suitable length of masking tape. The assembly and checkout are now complete.

Operating Hints and Suggestions

The mouse, used in conjunction with the intensity control on the monitor, is useful for varying the brightness of the video display. When the mouse is used with the contrast control, the user can adjust the luminous intensity ratio between the screen characters and the background. A little experimentation with the mouse will quickly demonstrate its other capabilities.

One caution--feline quadrupeds sometimes find the mouse interesting also

Some method was needed to hold the mouse between operating sessions. While chatting with the manager of our local hardware store, I mentioned the project and my need. He suggested something which appears to be almost designed for the job. He called it a "trap." That seems to be as good a name as any. The trap can be fastened to the monitor, the wall, or even to your computer table.

Future Trends

One industrial espionage agent, whom I have done some business with in the past, furnished me with a photograph he took in the secret research and development laboratories of a major computer manufacturer. Reportedly, their new interface will be named the "Hippo." One distinguishing feature of this advanced controller will be the fact that it is wireless. Presumably it communicates with the computer via infra-red or uses some kind of R.F. link.

I hope you will have as much fun building and using the mouse as I did. After the novelty wears off, it can always be used as a decoy for your next mouse hunt!

Robert Martin may be corresponded with at 45 Salem Lane, Little Silver, NJ 07739, or by ham radio at WB2KTG.

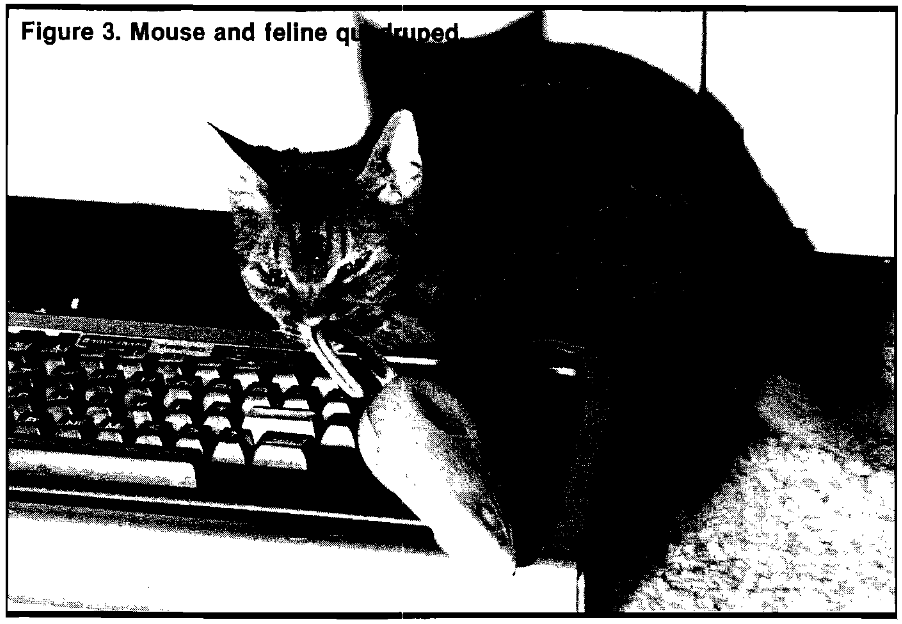


Figure 3. Mouse and feline quadruped.

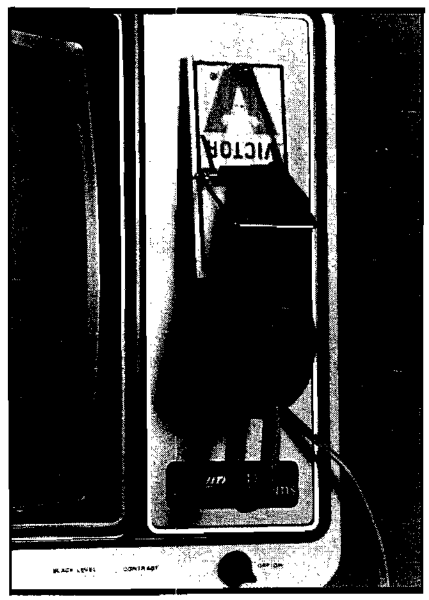


Figure 4. Mouse and mouseholder, sometimes called "trap."

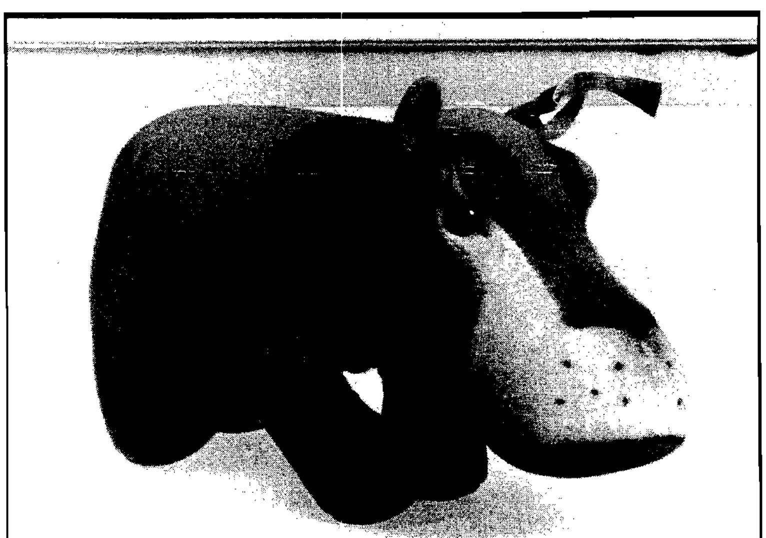


Figure 5. Secret photograph of possible successor to mouse, alias Hippo, an advanced wireless controller.

Double Vision Catalog
Double Vision Catalog
for
for
40 or 80 Columns
40 or 80 Columns

by Alan and Valerie Floeter
by Alan and Valerie Floeter

As more people buy Apple IIe's, we will be seeing more software using 80 columns. It is something to be expected. One logical usage of an 80 column width is in the CATALOG program, using the other half of the screen for listing file information. In the past, several DOS patches have appeared in publications to display two columns of file information for the normal 40 column display. Although we generally use short filenames, (we hate typing), the patches did shorten the amount of information sent to the screen. Now with a full 80 columns available, the complete file information can fit in two columns of 40 characters each.

When you think of it, most people's printers already have 80 or 132 columns, so why not have the CATALOG use all the available space, whether there are 40, 80, or 120. Why not take it even one step further and allow the CATALOG to shorten the filenames when wanted, printing multiple columns of files?

With these dreams, we set out to work on the CATALOG command for DOS. We were not only successful in the results we produced, but accomplished them with a patch that merely replaces the original CATALOG. This saved valuable patch space needed for other DOS enhancements already published.

So exactly what does this CATALOG patch do? Well,it depends on the number of characters your output device has and whether you use it in the normal or shortened mode. The chart in Figure 1 summarizes this.

Figure 1: Number of columns of filenames displayed

	COLUMNS 40	80	120
Normal	1	2	3
Shortened	2	4	6

If you use the normal CATALOG mode, 40 characters of information are displayed per filename. In the shortened mode, 20 characters are displayed per file,since the end of the filename is chopped off. If you send a CATALOG to an Apple II and Apple II Plus screen (40 column width), you'll see either one or two files per line depending on the mode. If you sent a CATALOG to an Apple IIe (80 column width), or an 80 column printer, such as an EPSON MX-80, you will see either 2 or 4 column files. Many printers have at least 120 characters per line, enabling them to produce 3 or 6 columns of file information.

How Was This Done?

When the Apple screen receives 40 characters, it automatically does a carriage return and line feed. We used this feature when we developed our CATALOG patch to DOS. Since the screen will take care of its own carriage returns, we just keep sending it information without telling it where the next line starts. This way the patch doesn't have to know how many columns the device has available. This CATALOG command sends out either 20 or 40 characters per file name continuously, and the printer or screen decides how much will fit on each line.

This works well for screens, but some printers or their interface cards might not be set up to send out a carriage return when their line is full. There is usually a switch on the printer or the interface to do this, or some specified control sequence will accomplish this. If you are unsure as to how your system handles this, just follow the suggestions we will give later.

One nice side benefit of this patch is that you don't need to do any POKE's to set up the number of columns. When you send out a CATALOG listing to two devices at the same time you will get different listings. For example, if you have a 132 column printer and an 80 column card and then enter "CATALOG", your screen will show two columns of filenames, while your printer will produce three columns.

Entering the Patch

We have written the assembly language routine to patch DOS for you. Enter the program, either into an assembler, or enter the opcodes, and save it to disk. Whenever you wish to have this patch in your DOS, BRUN the program. This could be part of your HELLO routine.

How to Shorten Filenames

When you want to shorten the filenames to store more information on the screen, enter POKE 44561,10 and POKE 44592,2. This will print 20 characters per file. To reset it back to the full 40 characters per file, enter POKE 44561,29 and POKE 44592,3.

DOS Warning

We always like to warn people about using a patched DOS. This patch doesn't use any of the patch space used by some of the other DOS improvements, so you shouldn't have any conflicts with other patches, but we can't guarantee it. Although we haven't had any problems, whenever you change a standard you can't predict if someone else assumed that part would stay the same.

Conclusion

Now you can utilize the entire line for CATALOG's, whether it is 40, 80 or 120 columns. Not only will you make better use of your display area, but you won't have to tell your device how many columns you have.

Figure 2. Sample CATALOGs for 40 and 80 columns.

Normal 40 Column Screen

```
CATALOG
DISK VOLUME 254
 A 002 HELLO
 T 001 PRINTER
 A 003 TEXT-TO-FOCUS
 A 003 FOCUS-TO-TEXT
 A 119 BUCHANAN
 A 003 FOCUS NEW
 A 039 BUCH-2
 B 054 MASTER DIRECTORY.L
 A 003 NEW FOCUS
 B 008 MASTER DIRECTORY
 B 009 MASTER DIR/DISPLAY.L
 T 001 HILISTER.MS
 T 008 BOOTMON2
 A 027 FLYNN
 A 027 MICRO MLR
 B 006 FLOETER
 A 041 BUCH1
]
```

Normal 80 Column Printer

```
DISK VOLUME 254
 A 002 HELLO                                 T 001 PRINTER
 A 003 TEXT-TO-FOCUS                         A 003 FOCUS-TO-TEXT
 A 119 BUCHANAN                              A 003 FOCUS NEW
 A 039 BUCH-2                                B 054 MASTER DIRECTORY.L
 A 003 NEW FOCUS                             B 008 MASTER DIRECTORY
 B 009 MASTER DIR/DISPLAY.L                  T 001 HILISTER.MS
 T 008 BOOTMON2                              A 027 FLYNN
 A 027 MICRO MLR                             B 006 FLOETER
 A 041 BUCH1
```

Shortened 80 Column Printer

```
CATALOG
DISK VOLUME 254
 A 002 HELLO          T 001 PRINTER       A 003 TEXT-TO-FOC    A 003 FOCUS-TO-TE
 A 119 BUCHANAN       A 003 FOCUS NEW     A 039 BUCH-2         B 054 MASTER DIRE
 A 003 NEW FOCUS      B 008 MASTER DIRE   B 009 MASTER DIR/    T 001 HILISTER.MS
 T 008 BOOTMON2       A 027 FLYNN         A 027 MICRO MLR      B 006 FLOETER
 A 041 BUCH1
]
```

Listing 1

```
                        ; PATCH FOR 80 COLUMN CATALOG FOR DOS 3.3
                        ;
                        ; BY AL FLOETER
                        ;
                        ;
0300                           ORG $300
                        ;
0085                    IND    EQU $85
0087                    COUNT  EQU $87
                        ;
0300 A2 00                     LDX #0
0302 BD 26 03           LOOP   LDA PATCH,X
0305 F0 1E                     BEQ DONE
0307 85 86                     STA IND+1       ; GET HI ORDER
0309 E8                        INX
030A BD 26 03                  LDA PATCH,X
030D 85 85                     STA IND         ; GET LO ORDER
030F E8                        INX
0310 BD 26 03                  LDA PATCH,X     ; GET COUNT
0313 85 87                     STA COUNT
0315 E8                        INX
0316 A0 00                     LDY #0
```

Listing 1
(continued)

```
0318 BD 26 03   DOIT    LDA PATCH,X     ; GET BYTE OF PATCH
031B 91 85              STA (IND),Y     ; PUT IT WHERE IT G
OES
031D E8                 INX
031E C8                 INY
031F C6 87              DEC COUNT       ; DONE YET
0321 D0 F5              BNE DOIT        ; NO
0323 F0 DD              BEQ LOOP        ; YES
0325 60         DONE    RTS
               ;
               ; PATCH AREA IN ORDER OF:
               ;  ADDRESS OF WHERE TO PUT IT
               ;  NUMBER OF BYTES TO MOVE
               ;  PATCH ITSELF
               ;
0326 AD A8      PATCH   BYT $AD,$A8
0328 06                 BYT 6
0329 EA                 NOP
032A A9 8D              LDA #$8D        ; OUTPUT CR
032C 20 ED FD           JSR $FDED
               ;
032F AD C3              BYT $AD,$C3
0331 06                 BYT 6
0332 EA                 NOP
0333 A9 8D              LDA #$8D        ; OUTPUT CR
0335 20 ED FD           JSR $FDED
               ;
0338 AD EB              BYT $AD,$EB
033A 30                 BYT $30
033B A0 07              LDY #7
033D 0A         SHIFT   ASL             ; FIND A BIT
033E 30 03              BMI GOTIT
0340 88                 DEY
0341 D0 FA              BNE SHIFT
0343 B9 A7 B3   GOTIT   LDA $B3A7,Y     ; GET PROGRAM TYPE
0346 20 ED FD           JSR $FDED       ; OUTPUT IT
0349 A9 A0              LDA #$A0        ; SPACE
034B 20 ED FD           JSR $FDED
034E BD E7 B4           LDA $B4E7,X
0351 85 44              STA $44
0353 BD E8 B4           LDA $B4E8,X     ; GET SIZE
0356 85 45              STA $45
0358 20 42 AE           JSR $AE42       ; OUTPUT IT
035B A9 A0              LDA #$A0        ; SPACE
035D 20 ED FD           JSR $FDED
0360 A0 1D              LDY #$1D        ; FILE NAME SIZE($A
 FOR 4 ACRO
0362 BD C9 B4   OLDOP   LDA $B4C9,X     ; GET NAME
0365 C9 A0              CMP #$A0        ; CONTROL CHARACTER
0367 B0 02              BCS NO
0369 A9 AA              LDA #$AA        ; MAKE IT A ASTERIS
K
               ;
036B AE 21      NO      BYT $AE,$21
036D 01                 BYT 1
036E F0                 BYT $F0
               ;
036F AE 2F              BYT $AE,$2F
0371 05                 BYT 5
0372 A2 03              LDX #3          ; 3 SPACES(2 FOR FO
UR ACROSS)
0374 20 4A F9           JSR $F94A
0377 00                 BYT 0
0378                    END
```

A New Variation on an Old Theme: *Replace Your 6502*

by Ron M. Battle

There are probably a lot of computer enthusiasts out there who have waited for a high-performance successor to the trusty 6502. Enter Rockwell's new CMOS 6502 product line. Although not the ultimate successor to the "old" 6502, the R65C00 family has many enhancements you might find quite interesting.

New Features

This new family of CMOS microprocessors comes in 3 models:

R65C02
R65C102
R65C112

All three will be available with your choice of operating frequency:

2 MHz
3 MHz
4 MHz

Each uses a single 5 volt ± 20% power supply drawing only 4 mA per MHz. By stopping the input clock, the processor will go into a standby mode and dissipate only 10 uW of power. In addition, 12 new instructions are added to the instruction set plus 2 new addressing modes.

Processor Description

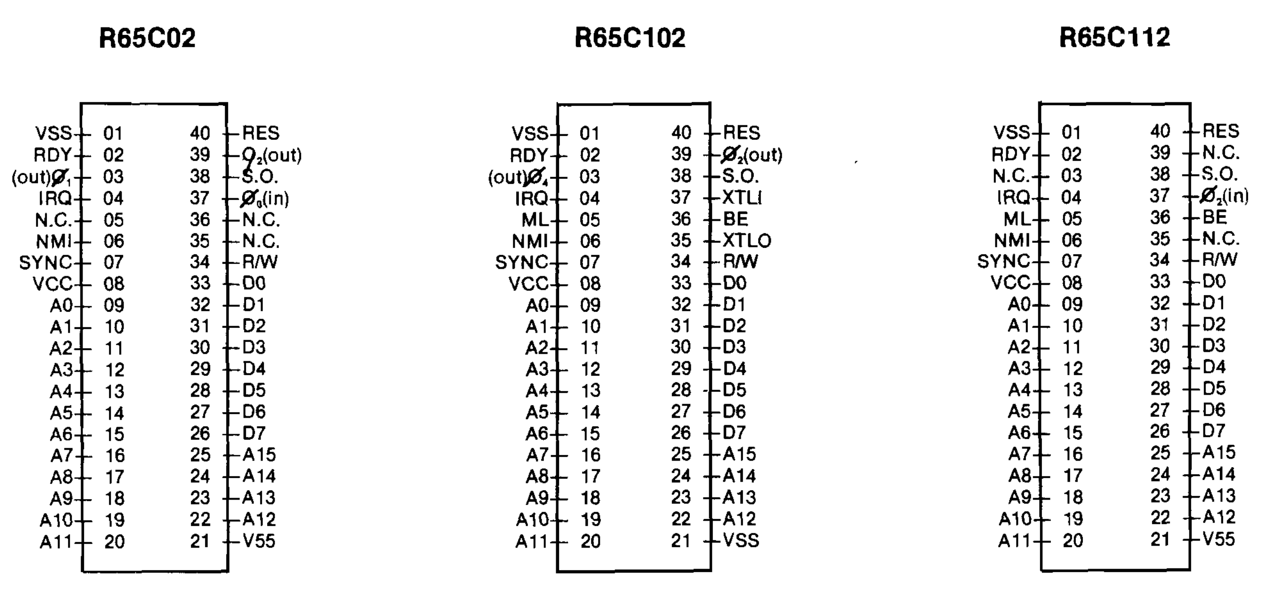
Figure 1 has the pinouts of these new chips and, as you can see, the R65C02 and R65C102 are pin compatible with the 6502.

R65C02: This is a direct replacement for the 6502.

R65C102: This new chip has functions on pins not used by the original 6502. No external time base is needed when a crystal is connected between pins 35 and 37, but the crystal frequency will be divided by four. Alternatively, you can input a TTL level single phase clock signal to pin 37 (XTLI) for compatibility with the 6502. Pin 3 (Phase 4) is a quadrature clock output used for peripheral timing. This output clock replaces the Phase 1 on the 6502. Pin 4 (ML), memory lock, is an output used by arbitration circuitry so read-modify-write instructions are not interrupted by external devices. Pin 36 (BE), bus enable, allows an external device to tri-state the data, address, and R/W lines by pulling this pin low. The R65C102 would be an interesting substitute for the 6502 so direct memory access (DMA) devices could be implemented easily.

R65C112: Designed as a slave processor, this model is used in conjunction with the R65C102 for a master-slave configuration. Pin 37 (Phase 2) is the input clock derived from the R65C102 Phase 2 output. This

Figure 1



chip has DMA capability like the R65C102.

New Addressing Modes

Indexed Absolute Indirect: This new 3 byte instruction takes 6 machine cycles to execute. The new opcode is C7H and new mnemonic is JMP(IND),X. In execution, the contents of the second and third bytes are added to the X register. The effective address is pointed to by this 16 bit result. This addressing mode comes in handy when you don't have room in zero page for a table of jump vectors or if you have a table of jump vectors in Read Only Memory (ROM).

Indirect: This new 2 byte instruction takes 5 or 6 machine cycles to execute. The second byte of this instruction is a zero page address. The zero page address points to the effective address, stored as low byte first, then high byte. This new addressing mode works with instructions ORA, AND, EOR, ADC, STA, LDA, CMP, and SBC.

New Instructions

Table 1 gives an overview of the new instructions. Most notable of these are the bit manipulation instructions. Most of these work on zero page bytes so that individual bits can be set (1) or reset (0), and program branching can be controlled by the status of each bit. These instructions facilitate coding for microprocessor based controller applications. The BRA, branch always instruction, is a handy tool for designing relocatable code and saves memory and machine cycles. The PHX, PHY, PLX, PLY instructions save memory and machine cycles, especially when used for interrupt processing. The STZ, store zero instruction, simplifies coding and will also save memory and machine cycles compared with alternate techniques.

Table 1

Mnemonic	Function
BBR	Branch on Bit Reset
BBS	Branch on Bit Set
BRA	Branch Always
PHX	Push X Register on Stack
PHY	Push Y Register on Stack
PLX	Pull X Register from Stack
PLY	Pull Y Register from Stack
RMB	Reset Memory Bit
SMB	Set Memory Bit
STZ	Store Zero
TRB	Test and Reset Bits
TSB	Test and Set Bits

Peripheral Support

To round out the R65C00 family, Rockwell has also introduced two CMOS peripheral chips, the R65C21 PIA and R65C24 PIA with timer. Both are low power versions of the 6521 Peripheral Interface Adapter which offers the user two 8-bit ports with handshaking. In addition, the R65C24 has a 16 bit timer on board for use in timing applications. Initially, two versions will be offered, 1 MHz and 2 MHz.

Things To Come

A high performance microcomputer system could be built using the 4 MHz R65C102, a DMA controller and fast arithmetic processor chip. With its bit manipulation instructions and low power consumption, the R65C02 could make a high performance controller utilizing FORTH. With the read access time of the 4 MHz processor being 168 nanoseconds, it will work with the newer 6116 CMOS 150 ns. memory chips. In fact, my next project is converting my Ohio Scientific 2 MHz micro to 4 MHz operation with the R65C102. Talk about computing in the FAST land!!!

For more information on the R65C00 family, contact: Rockwell International, Electronic Devices, P.O. Box C, Newport Beach, CA 92660; 714/833-4700.

Spec Box: 16K CoCo

Mike Hamilton

Requirements:
Any TRS-80C Color Computer

Have you ever wished you could have Extended BASIC for one day, just to see what high-resolution was like? Whether in text or high-resolution, the many modes of the Color Computer are generated by programs controlling the VDG (Video Display Generator). The graphics of Extended BASIC are programs stored in the Extended ROM chip. The program for high-resolution can easily be written in BASIC or assembly language, as illustrated by the following programs.

Three programs are provided, each accomplishing the same thing: high-resolution. **FAST** is for those that like fast, complex programs written in assembly language. LOADER is a BASIC utility program that loads and stores the **FAST** machine-code data. The second program, **SLOW**, is written in pure and simple, but slow, BASIC. The third, and my favorite, is HYBRID, combining the simplicity of BASIC and the speed of machine-code to create an efficient compromise. Use whichever you prefer, or use them all! Please note that only **HYBRID** contains a demonstration of the high-resolution.

How it Works

Certain steps must be taken to program the VDG for proper functioning. Each distinct step is documented in all the programs to help in understanding.

The first step in programming the VDG is reserving memory. This can be done via the clear statement. Line 10 of each program **'clears'** the required amount of memory. Since **FAST** is actually entered by LOADER, it does not require its own statement to reserve memory.

The second step is setting the proper values to the appropriate registers. This is the subroutine labeled **'PMODE 4'** in lines 700-800 of **FAST**, 1000-1030 of **SLOW**, and 10000-10030 of **HYBRID**.

The third is clearing the screen or video memory. Lines 640-690 of **FAST**, 10040 of **HYBRID**, and 1040 of **SLOW** accomplish this. As you can see in **SLOW**, this takes considerable time. **HYBRID** implements a machine-code subroutine that takes about 1/5 of a second. This is the only distinction between **SLOW** and **HYBRID**.

All that's left is plotting-setting, resetting, and pointing of a dot. Each can be implemented by various logical operations. Prior to the plotting in **SLOW** and **HYBRID**, the correct values of X and Y must be put in the X and Y variables. If, after calling the point subroutine, the variable PT is not equal to zero, then the point is set; else, it is reset.

FAST uses a slightly different approach. Before plotting, one must place the corresponding values of X and Y into Xval and Yval and set the SRP register. The SRP (set/reset/point) register must contain a zero to set, 255 to point, and any other value to reset. If, after calling the point subroutine, P reg contains 0, the point is reset; otherwise, a value of 255 means it's set.

Now you're ready to start experimenting with high-resolution. I recommend you use **HYBRID**, since it includes a demonstration and is considerably faster than **SLOW**. Try experimenting, such as changing the value of 248 to 240 in line 1020 of **SLOW** and line 10020 of **HYBRID**. If you're really ready to experiment, read section 4 of *Getting Started with Color BASIC*.

There are many other modes waiting to be used. Some are unavailable even through Extended BASIC, such as 192 x 64 resolution with 8 colors available at once. Good luck.

Mike Hamilton is a 15 year old computerist who lives in the small town of Checotah, Oklahoma, where the computer revolution is just starting. He has slightly over 3 years of programming experience and has never had a formal programming class. His equipment consists of an Extended Color Computer with 16K, a tape recorder, and small printer.

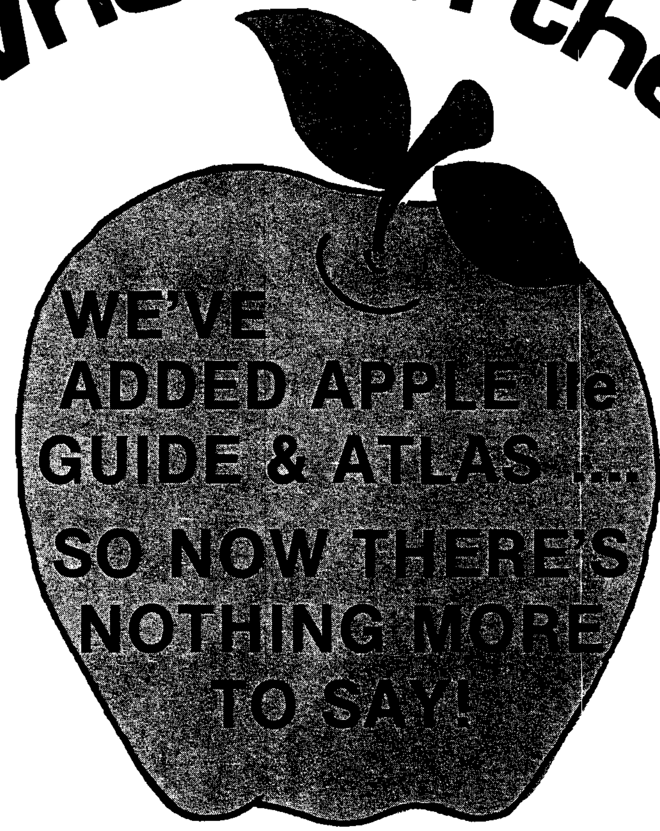
```
      1    HYBRID  USE ALL LINES EXCEPT ##
      2    SLOW    USE ALL LINES EXCEPT **
     10 CLEAR 255,10227  'RESERVE MEMORY
     19  ' READ AND STORE BIT PATTERNS
     20 FOR I=0 TO 7:READ A:VL(I)=A:NEXT I
**   29  'READ AND STORE CLS SUBROUTINE
**   30 FOR I=10228 TO 10239:READ A:POKE I,A:NEXT I
     35 GOSUB 10000:GOSUB 10040
     40 PI=3.14159:R=80
     50 FOR I=0 TO 2*PI STEP PI/330
     60 X=INT(R*SIN((90-I)*8)*SIN(I)+128)
     70 Y=INT(R*SIN((90-I)*1)*SIN(90-I)+96)
     80 GOSUB 10050: NEXT I
     90 GOTO 90
   9998  '  PMODE4 SUBROUTINE
   9999  'SET OFFSET VAL
  10000 POKE 65487,0:POKE 65483,0:POKE 65480,0
  10009  'SET VDG REGISTER
  10010 POKE 65472,0:POKE 65475,0:POKE 65477,0
  10019  'SET CONTROL REGISTER
  10020 POKE 65314,(PEEK(65314) AND 7) OR 248
```

```
   10030 RETURN
   10039  'PCLS SUBROUTINE
** 10040 EXEC 10228: RETURN
## 10040 FOR I=10240 TO 16383: POKE I,0:NEXT I:RETURN
   10049  'SET SUBROUTINE
   10050 GOSUB 10080:POKE LC,PEEK(LC) OR VL:RETURN
   10059  'RESET SUBROUTINE
   10060 GOSUB 10080:POKE LC,(PEEK(LC) AND (NOT VL)):
         RETURN
   10069  'POINT SUBROUTINE
   10070 GOSUB 10080:PT=PEEK(LC) AND VL:RETURN
   10080 LC=10240+(Y*32)+INT(X/8)
   10085 VL=VL(7-(X-INT(X/8)*8)): RETURN
   10090 DATA 1,2,4,8,16,32,64,128
** 10095 DATA 79,142,40,0,167,128,140,64,1,38,249,57
```

```
                   *  MIKE HAMILTON
                   *  HYBRID
                   *  APRIL/MAY 1984 ISSUE 71
                   *
                   * SYSTEM EQUATES
                   *
          27FA  LOC     EQU    10234
          27FC  SRP     EQU    10236
          27FD  PREG    EQU    10237
          27FE  XVAL    EQU    10238
          27FF  YVAL    EQU    10239

276A                    ORG    $276A

276A B6   27FF PLOT     LDA    YVAL   CHECK FOR INVALID CODE
276D 81   BF            CMPA   #191
276F 22   3F            BHI    STOP   RETURN IF SO
2771 CC   2800          LDD    #10240
2774 FD   27FA          STD    LOC    STORE OFFSET
2777 F6   27FF          LDB    YVAL
277A 86   20            LDA    #32
277C 3D                 MUL           FIND Y BYTE
277D F3   27FA          ADDD   LOC
2780 FD   27FA          STD    LOC
2783 F6   27FE          LDB    XVAL
2786 C4   07            ANDB   #7     FIND INDEX VALUE
2788 4F                 CLRA
2789 1F   01            TFR    D,X
278B F6   27FE          LDB    XVAL
278E 54                 LSRB          FIND X BYTE
278F 54                 LSRB
2790 54                 LSRB
2791 F3   27FA          ADDD   LOC
2794 FD   27FA          STD    LOC
2797 A6   89 27C9       LDA    DATA,X FETCH BIT PATTERN
279B 34   02            PSHS   A      STORE ON STACK
279D BE   27FA          LDX    LOC    LOAD BYTE VALUE
27A0 B6   27FC          LDA    SRP    CHECK SET/RESET/POINT
27A3 27   0C            BEQ    SET    IF 0 THEN SET
27A5 81   FF            CMPA   #$FF
27A7 27   0F            BEQ    POINT  IF 255 THEN POINT

27A9 35   02   RESET    PULS   A      ELSE, RESET
27AB 43                 COMA
27AC A4   84            ANDA   ,X
27AE A7   84            STA    ,X
27B0 39        STOP     RTS

27B1 35   02   SET      PULS   A
27B3 AA   84            ORA    ,X
27B5 A7   84            STA    ,X
27B7 39                 RTS

27B8 35   02   POINT    PULS   A
27BA A4   84            ANDA   ,X
27BC 27   06            BEQ    OFF
27BE 86   FF            LDA    #$FF
27C0 B7   27FD          STA    PREG   STORE 255 IF ON
27C3 39                 RTS

27C4 4F        OFF      CLRA
27C5 B7   27FD          STA    PREG   STORE 0 IF OFF
27C8 39                 RTS

                   * BIT PATTERNS

27C9 80 40 20 10 DATA   FCB    128,64,32,16,8,4,2,1
27CD 08 04 02 01

27D1 4F        CLS      CLRA
27D2 8E   2800          LDX    #10240
27D5 A7   80   LOOP     STA    ,X+    CLEAR THE SCREEN RAM
27D7 8C   4001          CMPX   #16385 TEST FOR END
27DA 26   F9            BNE    LOOP   CONTINUE
27DC 39                 RTS           DONE

27DD B6   FF22 PMODE4   LDA    65314
27E0 B7   FFCF          STA    65487  SET OFFSET VAL
27E3 B7   FFCB          STA    65483  OFFSET
27E6 B7   FFC8          STA    65480  OFFSET
27E9 B7   FFC0          STA    65472  SET VDG REGISTER
27EC B7   FFC3          STA    65475  VDG REGISTER
27EF B7   FFC5          STA    65477  VDG REGISTER
27F2 84   07            ANDA   #7     SET CONTROL REGISTER
27F4 8A   F8            ORA    #248
27F6 B7   FF22          STA    65314
27F9 39                 RTS

                        END
```

```
  1 '  FAST  LOADER
 10 CLEAR 255,10089
 20 FOR I=10090 TO 10233: READ A: POKE I,A: NEXT I
 30 DATA 182,39,255,129,191,34,63,204,40,0,253,39
 40 DATA 250,246,39,255,134,32,61,243,39,250,253,39
 50 DATA 250,246,39,254,196,7,79,31,1,246,39,254,84
 60 DATA 84,84,243,39,250,253,39,250,166,137
 70 DATA 39,201,52,2,190,39,250,182,39,252,39,12,129
 80 DATA 255,39,15,53,2,67,164,132,167,132,57,53,2
 90 DATA 170,132,167,132,57,53,2,164,132,39,6,134
100 DATA 255,183,39,253,57,79,183,39,253,57,128,64,32
110 DATA 16,8,4,2,1,79,142,40,0,167,128,140,64,1,38
120 DATA 249,57,182,255,34,183,255,207,183,255,203
130 DATA 183,255,200,183,255,192,183,255,195,183,255
140 DATA 197,132,7,138,248,183,255,34,57
```

What's Where in the APPLE



This famous book now contains the most comprehensive description of firmware and hardware ever published for *the whole Apple II family*. A new section with guide, atlas and gazeteer now provides Apple IIe specific information.

- Gives names and locations of various **Monitor, DOS, Integer BASIC** and **Applesoft** routines and tells what they're used for
- Lists **Peeks, Pokes** and **Calls** in over 2000 memory locations
- Allows easy movement between BASIC and Machine Language
- Explains how to use the information for easier, better, faster software writing

This expanded edition is available at the new low price of only $19.95

For the 35,000 people who already own previous editions, the IIe Appendix is available separately for just $5.00.

Please send me:

_______ What's Where in the Apple @ $19.95 ea. _______
(Plus $2.00 per copy shipping/handling)

_______ Apple //e Appendix @ $5.00 ea. _______
(includes shipping charges)

Mass residents add 5% sales tax $ _______

Total Enclosed $ _______

For faster service
Phone 617/256-3649

Name _______

Address _______

City _______ State _______ Zip _______

Signature _______

☐ Check ☐ VISA ☐ MasterCard

Acct. # _______ Expires _______

MICRO, P.O. Box 6502, Chelmsford, MA 01824

Easy DOES-IT (Not DOSPLUS)

Part 4

by Michael Keryan

DOES-IT -- Add new utility functions to a Commodore 64 by use of the RESTORE key.

Editor's Note: It has been brought to our attention that Micro System Software, Inc. has manufactured a software package under the trademark of DOS PLUS since 1981. We wish to make it clear that the programs included in the four parts of this article are in no way related to that product and that, had we been aware of the product, we would not have allowed the use of the name. To avoid further infringement on the trademark in question, we have renamed the series "DOES-IT," because in almost any case, no matter what utility your C-64 needs, this program "does-it".

This article adds two new functions to the recently published utility program (improperly named DOSPLUS in previous issues). The first is a program that allows BASIC programs to be hidden under ROM and swapped with the currently active BASIC program. The second is a time and alarm routine.

This is the fourth in a series of articles in which a number of machine language utility programs have been added to a Commodore 64. To access these utilities, you press the RESTORE key, which generates a Non-Maskable-Interrupt. The next key pressed determines which utility program is to be run.

The series of programs reside in unused RAM starting at location \$C800. Called DOES-IT, they can be loaded and initialized at the same time as the DOS WEDGE (located at \$CC00). In addition to the permanent utilities, additional transient programs can be called in from 'hidden' RAM located in the same address space as the BASIC ROM (\$A000-\$BFFF) and executed at \$C000.

The framework was given to allow anyone experienced with machine language programming to add their own routines and assign unique keys to access them. However, what if you have a BASIC program that you would want instant access to? If we can tuck away machine language programs in hidden RAM, why can't we do the same thing with our BASIC programs?

Hidden BASIC Programs

From the viewpoint of the computer's memory cells, a BASIC program is not much different than a machine language program; they both consist of a lot of 8 bit binary numbers. The procedure illustrated here can be used to store any BASIC program into hidden memory, provided that it can fit into this area. The hidden program is pulled out by the RESTORE key, followed by the left arrow key. In addition to pulling out this program, the BASIC program currently in memory is transferred to the same area of hidden RAM--the two programs exchange places. Therefore, the RESTORE, left arrow sequence can be used to toggle between two completely different BASIC programs.

Listing 1 is a BASIC program called DIRECTORY.PRINT that we will use to demonstrate hiding BASIC programs. The program is quite handy for producing compact directory listings on the printer. The directory entries can be listed as-is (unsorted) or sorted in alphanumeric order. The number of columns for the listing can be changed from the default of 3 by changing line 1. Using three columns (with the compressed mode of a printer) allows the listing to be small enough to be cut out and taped to the front of a diskette jacket.

To store this program (or any other BASIC program), proceed as follows. First (with DOES-IT activated) load the

Listing 1

```
0 :REM        M. J. KERYAN  11-12-83
1 :REM        MICRO #71 - APRIL 1984
2 :REM
9 :WIDTH=3: WD=WI-1: REM WIDTH=COLUMNS
10 DATA"******************************
20 DATA"*                            *
30 DATA"*    DIRECTORY    PRINTER    *
40 DATA"*                            *
50 DATA"******************************
60 DATA"*                            *
65 DATA"* 'S' SORTED     'U' UNSORTED *
70 DATA"* 'N' NO PRINT  'Q' QUIT     *
80 DATA"*                            *
90 DATA"******************************
100 POKE 53280,13:POKE 53281,7:POKE 646,0: DIM B$(100)
110 FOR I = 1 TO 10: READ A$(I): NEXT
120 PRINT"{CLEAR,DOWN6}": J=0 :FOR I = 1 TO 100:
    B$(I)="": NEXT
130 FOR I = 1 TO 10: PRINT"      {RVS}"A$(I)"{RVSOFF}":
    NEXT
140 GET Q$: IF Q$>"" THEN 140
150 GET Q$: IF Q$="" THEN 150
160 IF Q$="Q" THEN CLOSE 15: PRINT"{CLEAR}": END
170 IF Q$<>"S" AND Q$<>"U" AND Q$<>"N" THEN 120
200 PRINT"{CLEAR}    {RVS,SPACE32,RVSOFF}": Z=-1
210 GOSUB 880
280 GET#1,A$,B$
290 GET#1,A$,B$
300 GET#1,A$,B$
310 C=0
320 IF A$<>"" THEN C=ASC(A$)
330 IF B$<>"" THEN C=C+ASC(B$)*256
340 Z$=MID$(STR$(C),2)
350 IF LEN(Z$)<1 THEN Z$="   "
360 IF LEN(Z$)=1 THEN Z$="  "+Z$
370 IF LEN(Z$)=2 THEN Z$=" "+Z$
380 B$(J)=" "+Z$+" "+CHR$(34)
390 LZ=0
400 GET#1,B$: IF ST<>0 THEN GO TO 510
410 IF B$<>CHR$(34) THEN 400
420 GET#1,B$: IF B$<>CHR$(34) THEN B$(J)=B$(J)+B$:
    LZ=LZ+1: GOTO 420
430 GET#1,B$: IF B$=CHR$(32) THEN 430
440 B$(J)=B$(J)+CHR$(34): IF LZ>15 THEN 460
450 FOR JZ=LZ TO 15: B$(J)=B$(J)+" ": NEXT JZ
460 C$=""
470 C$=C$+B$: GET#1,B$: IF B$<>""  THEN 470
480 SS=ST: B$(J)=B$(J)+LEFT$(C$,3)
490 PRINT "    {RVS}  "B$(J)"    {RVSOFF}": J=J+1
500 IF SS=0 THEN 290
510 B$(J)=LEFT$(B$(J),LEN(B$(J))-1)+" BLOCKS FREE."
520 PRINT "    {RVS}  "B$(J)"              {RVSOFF}":
    CLOSE 1
530 IF Q$<>"N" THEN 570
540 GET Q$: IF Q$>"" THEN 540
550 GET Q$: IF Q$="" THEN 550
560 GO TO 120
570 IF Q$="U" THEN 680
580 LB=2
590 FOR II=J-1 TO LB STEP -1
600 IF MID$(B$(II-1),7,15) <= MID$(B$(II),7,15) THEN 650
610 EX$ = B$(II)
620 B$(II) = B$(II-1)
630 B$(II-1) = EX$
640 FX = II
650 NEXT II
660 IF LB=FX+1 THEN 680
670 LB = FX+1: GO TO 590
680 OPEN 4,4
690 PRINT#4: REM PRINT THE HEADER
700 PRINT#4,B$(0)
710 REM THE NEXT COMMANDS SENDS CONTROL TO PROWRITER
715 REM OR NEC-8023 PRINTERS THRU TYMAC 'CONNECTION'
720 REM TO SWITCH TO  CONDENSED MODE AND WIDE LINES
730 IF WI>2 THEN PRINT#4,CHR$(27)CHR$(27)"Q";:
    REM CONDENSED MODE FOR > 2 COL.
740 IF WI>3 THEN PRINT#4,CHR$(27)"W"CHR$(132);:
    REM WIDE LINES TO PREVENT CR'S
750 REM NOW PRINT THE DIRECTORY
760 RW = INT((J + WD)/ WI)
770 FOR I=1 TO RW: FOR W=1 TO WI
780 IF (I + (W-1)*RW ) > J THEN PRINT#4,"{SPACE26}";:
    GOTO 800
790 PRINT#4,B$(I + (W-1)*RW );
800 NEXT W: PRINT#4
810 NEXT I
820 PRINT#4
830 REM SWITCH PRINTER TO NORMAL
840 IF WI>2 THEN PRINT#4,CHR$(27)CHR$(27)"N";:
    REM UNCONDENSED
850 IF WI>3 THEN PRINT#4,CHR$(27)"W"CHR$(80);:
    REM BACK TO 80 COLUMN LINE
860 CLOSE 4
870 GO TO 120
880 CLOSE 15: OPEN 15,8,15
890 OPEN 1,8,0,"$0"
900 INPUT#15,E1,E2$,E3,E4
910 IF E1>0 THEN PRINT "{CLEAR,DOWN10,SPACE12}"E2$:
    CLOSE 1: GO TO 930
920 RETURN
930 GET W$: IF W$>"" THEN 930
940 GET W$: IF W$="" THEN 940
950 GO TO 890
```

Listing 2

```
C000                         ORG $C000
                   ;
0001               R6510     EQU $01        ;ROM SWITCH
002D               VARTAB    EQU $2D        ;VAR. POINTER
00C6               NDX       EQU $C6        ;KBD BF COUNT
0277               KEYD      EQU $0277      ;KBD BUFFER
CB41               MESSAG    EQU $CB41      ;MESSAGE PRINT
CBBD               BFLAG     EQU $CBBD      ;FLAG
                   ;
C000 AD BE CB      BASWAP    LDA BFLAG+1    ;IS FLAG=0?
C003 C9 00                   CMP #$00
C005 D0 14                   BNE SPECL      ;NO, BRANCH
C007 A6 2D                   LDX VARTAB     ;YES, ORIG PGM
C009 8E BD CB                STX BFLAG      ;VAR POINTER
C00C A6 2E                   LDX VARTAB+1   ;SAVE IT
C00E 8E BE CB                STX BFLAG+1
C011 E0 14                   CPX #20        ;<20?
C013 B0 15                   BCS SWAPB      ;NO, LEAVE PNTR
C015 A9 14                   LDA #20        ;YES, EXPAND
```

```
C017 85 2E                STA VARTAB+1
C019 D0 0F                BNE SWAPB      ;BRANCH ALWAYS
C01B AE BD CB   SPECL     LDX BFLAG      ;GET BACK FLAG
C01E 86 2D                STX VARTAB     ;RESTORE PNTR
C020 AE BE CB             LDX BFLAG+1
C023 86 2E                STX VARTAB+1
C025 A9 00                LDA #$00       ;SET FLAG
C027 8D BE CB             STA BFLAG+1
C02A A9 A7      SWAPB     LDA #$A7       ;MAKE SURE
C02C 8D 47 C0             STA BAS1+2     ;THAT ROUTINE
C02F 8D 4E C0             STA BAS2+2     ;IS SET-UP IN
C032 A9 08                LDA #$08       ;CASE OF REENTRY
C034 8D 4B C0             STA RAM1+2
C037 8D 52 C0             STA RAM2+2
C03A 78                   SEI            ;NOW SWAP
C03B A5 01                LDA R6510
C03D 29 FE                AND #$FE
C03F 85 01                STA R6510
C041 A0 0B                LDY #11        ;11 BLOCKS
C043 A2 00                LDX #$00
C045 BD 00 A7   BAS1      LDA $A700,X
C048 48                   PHA
C049 BD 00 08   RAM1      LDA $0800,X
C04C 9D 00 A7   BAS2      STA $A700,X
C04F 68                   PLA
C050 9D 00 08   RAM2      STA $0800,X
C053 E8                   INX
C054 D0 EF                BNE BAS1
C056 EE 47 C0             INC BAS1+2
C059 EE 4E C0             INC BAS2+2
C05C EE 4B C0             INC RAM1+2
C05F EE 52 C0             INC RAM2+2
C062 88                   DEY
C063 D0 E0                BNE BAS1
C065 A5 01                LDA R6510      ;RESTORE BASIC
C067 09 01                ORA #$01
C069 85 01                STA R6510
C06B 58                   CLI
C06C 20 41 CB             JSR MESSAG
C06F 93                   BYT $93
C070 20 20 20             ASC '              PRESS
C081 12                   BYT $12
C082 52 45 54         ASC 'RETURN'
C088 92               BYT $92
C089 20 54 4F         ASC ' TO RUN'
C090 0D 43 4C         BYT $0D,$43,$4C,$52,$0D
C095 11 11 52         BYT $11,$11,$52,$55,$4E,$91,$91,$0
C09D A9 20            LDA #$20
C09F 8D 77 02         STA KEYD
C0A2 A9 20            LDA #$20
C0A4 8D 78 02         STA KEYD+1
C0A7 A9 20            LDA #$20
C0A9 8D 79 02         STA KEYD+2
C0AC A9 13            LDA #$13       ;STORE SOME
C0AE 8D 7A 02         STA KEYD+3     ;STUFF IN THE
C0B1 A9 11            LDA #$11       ;KEYBOARD
C0B3 8D 7B 02         STA KEYD+4     ;BUFFER
C0B6 A9 0D            LDA #$0D
C0B8 8D 7C 02         STA KEYD+5
C0BB A9 06            LDA #$06
C0BD 85 C6            STA NDX
C0BF 60               RTS
                  ;
C0C0                  END
```

Listing 3

```
0340                      ORG $0340
                ;
0314            CINV      EQU $0314
0420            TIMDIS    EQU $0420
D020            BORDER    EQU $D020
0286            COLOR     EQU $0286
D418            SIDVOL    EQU $D418
D820            DISCLR    EQU $D820
DC08            TENTHS    EQU $DC08
DC09            SECS      EQU TENTHS+1
DC0A            MINS      EQU TENTHS+2
DC0B            HOURS     EQU TENTHS+3
DC0D            CIAINT    EQU TENTHS+5
                ;
0340 AD 0D DC   TIMIRQ    LDA CIAINT
0343 29 04                AND #$04
0345 F0 03                BEQ BEGIN
0347 8D FA 03             STA ALFLAG
034A AD FA 03   BEGIN     LDA ALFLAG
034D F0 20                BEQ DISTIM
034F 8D F9 03             STA DISPFL
0352 A5 A2                LDA $A2
0354 6A                   ROR A
0355 6A                   ROR A
0356 6A                   ROR A
0357 29 0C                AND #$0C
0359 8D 20 D0             STA BORDER
035C 29 04                AND #$04
035E 8D 18 D4             STA SIDVOL
0361 A5 C5                LDA $C5
0363 C9 04                CMP #$04
0365 D0 08                BNE DISTIM
0367 A2 00                LDX #$00
0369 8E FA 03             STX ALFLAG
036C 8E F9 03             STX DISPFL
036F AD F9 03   DISTIM    LDA DISPFL
0372 F0 60                BEQ TIMRET
0374 AD 0B DC             LDA HOURS
0377 AA                   TAX
0378 29 0F                AND #$0F
037A 18                   CLC
037B 69 30                ADC #$30
037D 8D 21 04             STA TIMDIS+1
0380 8A                   TXA
0381 10 04                BPL LBLA
0383 A2 10                LDX #$10
0385 10 02                BPL LBLB
0387 A2 01      LBLA      LDX #$01
0389 8E 26 04   LBLB      STX TIMDIS+6
038C A2 20                LDX #$20
038E 29 10                AND #$10
0390 F0 02                BEQ LBLC
0392 A2 31                LDX #$31
0394 8E 20 04   LBLC      STX TIMDIS
0397 AD 0A DC             LDA MINS
039A AA                   TAX
039B 29 0F                AND #$0F
039D 69 30                ADC #$30
039F 8D 24 04             STA TIMDIS+4
03A2 8A                   TXA
03A3 4A                   LSR A
```

Listing 3 *(continued)*

```
03A4 4A              LSR A
03A5 4A              LSR A
03A6 4A              LSR A
03A7 18              CLC
03A8 69 30           ADC #$30
03AA 8D 23 04        STA TIMDIS+3
03AD A9 3A           LDA #$3A
03AF 8D 22 04        STA TIMDIS+2
03B2 A9 0D           LDA #$0D
03B4 8D 27 04        STA TIMDIS+7
03B7 A2 20           LDX #$20
03B9 8E 25 04        STX TIMDIS+5
03BC AD 09 DC        LDA SECS
03BF 29 01           AND #$01
03C1 F0 03           BEQ STCLOK
03C3 8E 22 04        STX TIMDIS+2
03C6 AD 08 DC STCLOK LDA TENTHS
03C9 AD 86 02        LDA COLOR
03CC A2 07           LDX #$07
03CE 9D 20 D8 LBLE   STA DISCLR,X
03D1 CA              DEX
03D2 D0 FA           BNE LBLE
03D4 4C 31 EA TIMRET JMP $EA31
03D7 78       TMINIT SEI
03D8 AD 14 03        LDA CINV
03DB A2 40           LDX #<TIMIRQ
03DD 8D D5 03        STA TIMRET+1
03E0 8E 14 03        STX CINV
03E3 AD 15 03        LDA CINV+1
03E6 A2 03           LDX #>TIMIRQ
03E8 8D D6 03        STA TIMRET+2
03EB 8E 15 03        STX CINV+1
03EE 58              CLI
03EF 60              RTS
              ;
              ;TIME FUNCTION--TOGGLE DISPLAY
              ; ON AND OFF--ALARM REMAINS ACTIVE
03F0 A9 01    TIME   LDA #$01
03F2 4D F9 03        EOR DISPFL
03F5 8D F9 03        STA DISPFL
03F8 60              RTS
              ;
03F9 00       DISPFL BYT $00
03FA 00       ALFLAG BYT $00
              ;
03FB                 END
```

Listing 4

```
10 REM********************************
20 REM*                              *
30 REM*  DOS+ LOADER   M.J.KERYAN    *
40 REM*  MICRO #71 - APRIL 1984      *
50 REM* NEW FEATURES INCLUDE         *
60 REM* THE ML ROUTINES: REPEAT,     *
70 REM* DIRECT.PRINT, FORMAT PRINTER, *
80 REM* SETS UP TIME, SUPERMON, KILL *
90 REM********************************
900 IF PEEK(52159)=96 THEN GOTO 1040
1000 LOAD " D+++.ML",8,1
1040 SYS 51200
1050 IF A=1 THEN 1150
1100 PRINT"{CLEAR,DOWN2}WANT TO USE THE TIMER/ALARM CLOCK? (Y/N)":A=1
1110 GET AQ$:IF AQ$=""THEN 1110
1150 IF PEEK(2)=115 THEN GOTO 1400
1200 PRINT"{CLEAR,DOWN2}WANT TO USE SUPERMON":PRINT"AND/OR FORMAT PRINTER"
1205 PRINT"AND/OR PRINT DIRECTORY  (Y/N)"
1210 GET AS$:IF AS$=""THEN 1210
1220 IF AS$="N" THEN 1400
1230 POKE 2,115: LOAD " FPDPSM.ML",8,1
1400 REM******************************
1410 REM*                            *
1420 REM* USRHELP SCREEN  CALL BY     *
1430 REM*   RESTORE, U  OR SYS 51265 *
1440 REM*                            *
1450 REM******************************
1500 POKE 53280,0: POKE 53281,0
1510 PRINT"{GREEN,CLEAR,DOWN11}       {RVS}PLACE YOUR SCREEN HERE{RVSOFF}"
1520 POKE 53128,4: POKE 53131,232
1530 SYS 53164
1540 POKE 60392,PEEK(53280)
1550 POKE 60393,PEEK(53281)
1560 POKE 53128,216: POKE 53131,236
1570 SYS 53164
```

BASIC program into memory. Then save it to disk and get a directory listing (by pressing @ then $). Multiply the required number of blocks by 256, then subtract 1 (you should get 11x256 -1 2815). Next, in immediate mode, type the following:

```
FOR I=0TO2815: A=PEEK(I+2048): POKE
I+42752, A: NEXT
```

(If you got a number other than 2815, use it above.) Now load into memory at $C000 the one-block machine language program as shown in Listing 2. Use either an assembler, the monitor, or a BASIC loader that POKEs DATA into memory. If your required number of blocks was not 11, place your number at $C042, add nine and place this number at $C012 and at $C016. Then move this program to hidden RAM also:

```
FOR I=0TO255: A=PEEK(I+49152): POKE
I+42496,A: NEXT
    POKE 40991,1: POKE 41023,166
```

So far, we've moved the one-block boot program to $A600, the BASIC program starting at $A700, and set up table pointers for the left arrow function. Now press RESTORE, S to get into the monitor and type:

```
.S " FPDPSM.ML",08,A000,C000
```

This saves to disk the BASIC program as well as the other programs previously hidden (printer formatting and the monitor).

The machine language BASIC boot program deserves some explanation (refer to Listing 2). A flag (located at $CBBE) is initially set to zero. If it is zero, then the program BASWAP knows that this function hasn't yet been activated and our hidden BASIC program is still hidden. If so, the pointers to the beginning of the variables table (located at $002D, $002E) are saved in $CBBD, $CBBE. If the current BASIC program is larger than our hidden program, everything is OK, but if it is smaller, then we move the variable pointer table up to make room for the new program.

Next, the two basic programs are swapped (or at least the first 11 blocks of the current program is swapped in this example). A CLR command is activated; then the option is given to RUN the new program by pressing the RETURN key. This is accomplished by printing appropriate information on the screen and stuffing carriage returns into the keyboard buffer. A few spaces stuck into the buffer guard against multiple

Listing 4 *(continued)*

```
1900 REM********************************
1910 REM*                              *
1920 REM*  HELP SCREEN    CALL BY      *
1930 REM*   RESTORE, H  OR SYS 51232   *
1940 REM*                              *
1950 REM********************************
2000 POKE 53280,5: POKE 53281,1
2010 PRINT"{CLEAR}            {RED,RVS}COMMAND SUMMARY{RVSOFF}"
2020 PRINT"    DOS 5.1              DOES-IT 1.2"
2030 PRINT" ----------------   --------------------"
2040 PRINT"{BLACK} {UP ARROW}PGM{BLUE} LOAD & RUN    {BLACK}RESTORE
     {BLUE} STOP SCROLL"
2050 PRINT"{BLACK} /PGM{BLUE} LOAD PGM"
2060 PRINT"{BLACK} %PGM{BLUE} LOAD ML PGM         --FOLLOW BY:"
2070 IF AS$="N" THEN PRINT
2075 IF AS$<>"N"THEN PRINT"                    {GREEN,BACK ARROW,
     BLUE} PRINT DIRECTORY"
2080 PRINT"{BLACK} {BACK ARROW}PGM{BLUE} SAVE PGM      {BLACK}A
     {BLUE} APPEND PGMS"
2090 PRINT
2100 PRINT"{BLACK} @${BLUE}   LIST DIR      {BLACK}B{BLUE}  BACKGND COLOR"
2110 PRINT"                {BLACK}C{BLUE}  CHARACTER COLORS"
2120 PRINT"{BLACK} @N0:NAME,ID        E{BLUE}  EDGE COLORS"
2130 PRINT"      FORMAT DISK"
2135 IF AS$<>"N" THEN PRINT"{UP}      FORMAT DISK   {GREEN}F
     {BLUE} FORMAT PRINTER"
2140 PRINT"{BLACK} @R0:NEWNM=OLDNM    D{BLUE}  DUMP TO PRINTER"
2150 PRINT"      RENAME DISK   {BLACK}O{BLUE}  PRINTER OFF"
2160 PRINT"{BLACK} @C0:NEWPGM=OLDPGM  P{BLUE}  PRINTER ON"
2170 PRINT"      COPY PGM      {BLACK}R{BLUE}  REPEAT ON/OFF"
2180 PRINT"{BLACK} @S0:PGM            H{BLUE}  HELP (THIS LIST)"
2190 PRINT"      SCRATCH PGM   {BLACK}U{BLUE}  USER HELP"
2200 PRINT"{BLACK} @I{BLUE}   INITIALIZE"
2205 IF AQ$<>"N" THEN PRINT"{UP,BLACK} @I{BLUE}   INITIALIZE    {BLACK}T
     {BLUE}  TIMER/ALARM"
2210 PRINT"{BLACK} @UI{BLUE}  RESET          {BLACK}N{BLUE}  NUMBER CONVERSN."
2220 PRINT"{BLACK} @V{BLUE}   VALIDATE     "
2225 IF AS$<>"N" THEN  PRINT"{UP,BLACK} @V{BLUE}   VALIDATE      {GREEN}S
     {BLUE}  SUPERMON"
2230 PRINT"{BLACK} @Q{BLUE}   QUIT DOS 5.1  {BLACK}K{BLUE}  KILL ALL (RESET)"
2240 PRINT"                {BLACK}RETURN{BLUE}  CANCEL"
2250 PRINT"      {RED}PRESS {RVS}RETURN{RVSOFF} TO CONTINUE";
3000 POKE 53128,4: POKE 53131,248
3010 SYS 53164
3020 POKE 64488,PEEK(53280)
3030 POKE 64489,PEEK(53281)
3040 POKE 53128,216: POKE 53131,252
3050 SYS 53164
3055 IF AS$="N" THEN GOTO 3060
3056 POKE 52179,89:POKE 52211,207:POKE 52166,89:POKE 52198,207: POKE 52157,0
3057 POKE 52191,89:POKE 52223,207: POKE 52158,0: POKE 53123,0
3060 IF AQ$="Y" THEN GOSUB 5000
3065 GETB$:IF B$="" THEN 3065
3066 PRINT"{CLEAR}":IF AQ$="Y" THEN GOSUB 5030
3080 POKE 52171,226:POKE 52203,252
3090 POKE 53280,6: POKE 53281,12:POKE 646,0
3100 PRINT"{HOME,DOWN4}PRINT (FRE(0)+65535){DOWN,LEFT12}BYTES FREE":POKE 198,6
3110 POKE 631,19:POKE 632,17:POKE 633,17:POKE 634,17:POKE 635,17:POKE 636,13
3200 NEW
5000 FOR I=832 TO 1018:READ A:POKE I,A:NEXT I
5010 POKE 52180,240:POKE 52212,3:SYS 983
5020 POKE 51560,32:POKE 51561,215:POKE 51562,3
5025 RETURN
```

RESTORE key activations which sometimes occur when you press this key. If you don't want to RUN the program, press the cursor down key instead of the RETURN key.

To go back to the original BASIC program, hit RESTORE, left arrow again. Now the variable table pointers are restored (from $CBBD, $CBBE) and then the swap is performed again. This sticks our originally hidden program back in its hiding place and returns our original BASIC program to the BASIC workspace, unharmed.

Time Routine

In the December 1983 MICRO, a very nice machine language time/alarm routine was described. Unfortunately, this program is incompatible with DOES-IT's since they both use the same memory area at $02A7. Therefore, Ian Adam's program was shortened and revised somewhat to tie it into DOES-IT.

The revised program is shown in Listing 3. The entire program fits into the cassette buffer from $0340 to $03FB. The changes from the original program are as follows:

The RESTORE, T key sequence is used to toggle only the time display on and off. This allows the alarm to remain active even if the time is not displayed. No SYS calls are necessary.

Only hours, minutes, and AM/PM is shown in the upper right corner of the screen. The seconds and tenths of seconds proved to be distracting; they were replaced by a blinking colon to let you know the clock is still ticking.

The alarm function is nearly the same, except the word ALARM was left out to save space. Function key F1 turns off both the alarm and the display (but you can reactivate the display with RESTORE, T).

The characters used in the time display always use the currently active character color. This assures visibility. A warning: never leave the time display on the screen when you are editing programs. It is very easy to accidently edit the current time of day into your BASIC programs. The time and alarm setting is done through the new DOES-IT boot program, DOS+.

Getting It All Together

The machine language routine DOES-IT.ML must be changed to incorporate the vectors for the Repeat and Kill functions from Part 3 (or the required

Listing 4 *(continued)*

```
5030 INPUT"{DOWN5} IS IT NOW {RVS} AM {RVSOFF} OR {RVS} PM
     {RVSOFF}";A$:INPUT"{DOWN} THE HOUR";H
5040 PRINT"{DOWN2} ENTER THE MINUTE TO START THE CLOCK"
5050 PRINT"{DOWN}THE CLOCK WILL START WHEN YOU HIT {RVS}RETURN{RVSOFF,
     DOWN} ENTER THE MINUTE";
5060 IF H>12 THEN A$="P":H=H-12:GOTO 5060
5070 IF H>9 THEN H=H+6
5080 IF LEFT$(A$,1)="P" THEN H=H+128
5090 C=56328:POKE C+3,H:POKE C+1,0
5100 INPUT M:M=M+INT(M/10)*6
5110 POKE C+2,M:POKE C,0:SYS 1008:PRINT"{DOWN2,RVS}    IF TIME NOT OK,
      PRESS ANY KEY    "
5120 FOR I=1 TO 1000:IF PEEK(198) THEN POKE 198,0:SYS 1008:GOTO 5030
5130 NEXT
5140 PRINT"{CLEAR,DOWN5} WHAT TIME WOULD YOU LIKE THE ALARM{DOWN}":
     INPUT"{RVS} AM {RVSOFF} OR {RVS} PM {RVSOFF}";A$
5150 A$=LEFT$(A$,1):INPUT"{DOWN} THE HOUR";H:HH=H
5155 IF H>12 THEN A$="P":H=H-12:GOTO 5155
5160 H=H-6*(H>9)-128*(A$="P"):INPUT"{DOWN} THE MINUTE";M:MM=M:M=M+INT(M/10)*6
5175 POKE C+7,136:POKE C+3,H:POKE C+2,M:POKE C,1:POKE C+7,8
5180 POKE 54273,99:POKE 54278,240:POKE 54276,21:POKE 54287,2:POKE 54290,17
5185 IF MM<10 THEN PRINT"{CLEAR,DOWN2}"SPC(27-HH/9.9);HH;"{LEFT}:";
     MM;A$"M ALARM":GOTO 5195
5190 PRINT"{CLEAR,DOWN2}"SPC(27-HH/9.9);HH;"{LEFT}:";
     RIGHT$(STR$(MM),2);" "A$"M ALARM"
5195 RETURN
6000 DATA 173, 13, 220, 41, 4, 240, 3, 141, 250, 3
6010 DATA 173, 250, 3, 240, 32, 141, 249, 3, 165, 162
6020 DATA 106, 106, 106, 41, 12, 141, 32, 208, 41, 4
6030 DATA 141, 24, 212, 165, 197, 201, 4, 208, 8, 162
6040 DATA 0, 142, 250, 3, 142, 249, 3, 173, 249, 3
6050 DATA 240, 96, 173, 11, 220, 170, 41, 15, 24, 105
6060 DATA 48, 141, 33, 4, 138, 16, 4, 162, 16, 16
6070 DATA 2, 162, 1, 142, 38, 4, 162, 32, 41, 16
6080 DATA 240, 2, 162, 49, 142, 32, 4, 173, 10, 220
6090 DATA 170, 41, 15, 105, 48, 141, 36, 4, 138, 74
6100 DATA 74, 74, 74, 24, 105, 48, 141, 35, 4, 169
6110 DATA 58, 141, 34, 4, 169, 13, 141, 39, 4, 162
6120 DATA 32, 142, 37, 4, 173, 9, 220, 41, 1, 240
6130 DATA 3, 142, 34, 4, 173, 8, 220, 173, 134, 2
6140 DATA 162, 8, 157, 31, 216, 202, 208, 250, 76, 49
6150 DATA 234, 120, 173, 20, 3, 162, 64, 141, 213, 3
6160 DATA 142, 20, 3, 173, 21, 3, 162, 3, 141, 214
6170 DATA 3, 142, 21, 3, 88, 96, 169, 1, 77, 249
6180 DATA 3, 141, 249, 3, 96, 0, 0
```

Figure 1. DOES-IT Help Screen

```
                COMMAND SUMMARY
     DOS 5.1                     DOES-IT
     ----------------      -----------------

^PGM LOAD & RUN    RESTORE STOP SCROLL
/PGM LOAD PGM
%PGM LOAD ML PGM          --FOLLOW BY:
                   ←  PRINT DIRECTORY
←PGM SAVE PGM      A  APPEND PGMS

@$   LIST DIR      B  BACKGND COLOR
                   C  CHARACTER COLORS
@N0:NAME,ID        E  EDGE COLORS
     FORMAT DISK   F  FORMAT PRINTER
@R0:NEWNM=OLDNM    D  DUMP TO PRINTER
     RENAME DISK   O  PRINTER OFF
@C0:NEWPGM=OLDPGM  P  PRINTER ON
     COPY PGM      R  REPEAT ON/OFF
@S0:PGM            H  HELP (THIS LIST)
     SCRATCH PGM   U  USER HELP
@I   INITIALIZE    T  TIMER/ALARM
@UI  RESET         N  NUMBER CONVERSN.
@V   VALIDATE      S  SUPERMON
@Q   QUIT DOS 5.1  K  KILL ALL (RESET)
                   RETURN  CANCEL
     PRESS RETURN TO CONTINUE
```

POKEs should be included in the loader program). However, it is desirable to keep the Time function and the transient programs as options so we can bypass loading and executing them if we desire. Therefore the loader program, now called DOS+, has been changed; see Listing 4.

If the machine language portion of DOES-IT from $C800 is not in memory, the loader program will load it (it is now called D+++.ML) as shown in lines 900-1000. A SYS to 51200 is made to initialize DOES-IT and the wedge. Then you are asked if you want the Time routine and the transient programs. If you answer yes to the second question, the 8K block FPDPSM.ML is loaded into hidden RAM at $A000-$BFFF, only if the flag (at location $0002) indicates that it is not yet in memory

The HELP screen was changed to reflect all the additions to DOES-IT and will only print out those additions that are actually loaded into memory. A printout of the latest HELP screen is shown in Figure 1 (but it looks better on a color monitor).

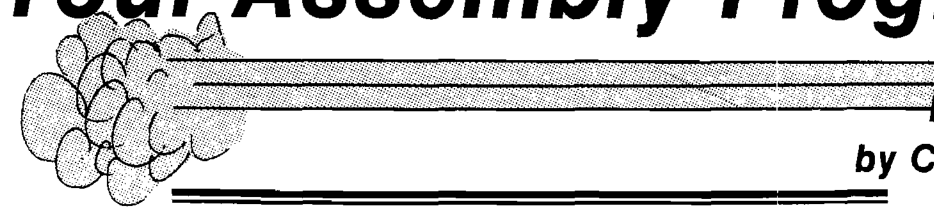
If the time routine is desired, then lines 5000 are run, in which the current time and the alarm time are entered. Ending the program leaves the time display on, the program NEWs itself, then prints the available free memory.

The DOES-IT routines now consist of three programs that can be copied to copies to your other disks: DOS--, the BASIC boot program, D+++.ML, the permanent ML programs and tables that load into $C800-$CFFF (this also contains the DOS WEDGE), and FPDPSM.ML, the 8K block that loads into hidden RAM ($A000-$BFFF). To activate DOES-IT, type in LOAD "DOS+",8 then RUN.

In general the routines are quite easy to use. However, entering them from the keyboard for the first time can be confusing due to the complexity of operations involved. For $10 (US), MICRO will provide the DOES-IT routines from the four articles in this series on disk, along with all the assembly source listings. For foreign requests, please send sufficient postage. For those hackers interested in adding more functions, five blocks of memory is unused in the hidden RAM from $B200-$B6FF, eight block are available from $E000-$E7FF, and all sixteen blocks are available in the hidden $D000-$DFFF area.

MICRO

Four Techniques to Make Your Assembly Programs FAST



by Chris Williams

These four 6502 assembly language programming techniques are designed with one and only one consideration in mind. Speed. Raw, unadulterated, eyebrow-raising speed.

These techniques are applicable to any 6502 microcomputer. They have nothing whatsoever to do with graphics or sound or anything else that would require special machine-specific arrangements. They are meant to be entirely generic. In fact, the underlying concepts are not limited to the 6502. They are readily applicable to all 8-bit machines.

A relentless pursuit of speed-of-execution is perhaps the noblest of activities for a programmer. While speed isn't the only characteristic of a fine program, it usually is the characteristic most difficult to achieve. As a result, when a programmer has a finished routine that absolutely screams through its task, he tends to grin a bit wider and finds himself a bit more anxious to show it off--especially to knowledgeable friends who will appreciate what they're seeing.

So, if you're sitting there now nodding at the familiarity of that scene, and if you suspect that your routines could execute faster, then you'd be well advised to study the following techniques carefully...and use them! Writing fast programs is a skill, not a talent, and skills are perfected through practice.

Counting Up or Counting Down

From the perspective of speed, choosing to count up is the single most common mistake in the typical assembly language program. If you're taking notes, write this down. *In general, counting up is slower than counting down.* Watch.

Suppose I have an application that requires an index to count through a list of values. Here's how the typical program does it:

```
     LDY #0      ;initialize index
LOOP LDA (LOC),Y ;get value from list
     STA PLACE   ;do something with it
     INY         ;increment index
     CPY MAXVAL  ;check to see if done
     BNE LOOP    ;not done, loop
DONE --- ----    ;else, proceed
```

Count the instructions. There's six, four for loop management, two (the LDA and STA) functional. That's about the best we can do counting up.

Instead, let's arrange things to count down.

```
     LDY MAXVAL  ;initialize index
LOOP LDA (LOC),Y ;get value from list
     STA PLACE   ;do something with it
     DEY         ;decrement index
     BNE LOOP    ;check for zero to
                  finish
DONE --- ----    ;else, proceed
```

Five instructions this time. No CPY, which is a four-cycle instruction (absolute addressing). So our loop here is four machine cycles faster than w: counting up.

This kind of thing is always true. You are never better off counting up and checking a count. If MAXVAL is 256, you can eliminate the CPY when counting up and achieve indentical speed, but MAXVAL is rarely 256. Count down!

Fast Double Precision

All 6502 microcomputers have a 64K memory maximum, assuming no bank switching. 64K is 65536 which is $FFFF hex. $FFFF hex cannot be represented by a single byte; it requires two.

What this all means is that any addressing routines you might need have to be double precision (assuming you're looking for more than just 256 bytes). Below is a fast, general technique for doing additive double-byte addressing. And below that is a special case method for doing the same thing even faster.

```
CLC
LDA  $LOC   ;get least significant byte
ADC  #VAL   ;add immediate value
STA  $LOC   ;store result in L.S. byte
LDA  $LOC+1 ;get M.S. byte
ADC  #0     ;add with carry zero
STA  $LOC+1 ;store in M.S. byte
RTS
```

The carry from the least significant byte operation flows into the most significant byte operation. This results in a nice, tight, double precision add.

If #VAL is equal to 1, as it often is, we can get even faster.

```
INC  $LOC      ;increment L.S. byte
BNE  OUT       ;RTS unless = 0, from
                previous FF
INC  $LOC+1    ;overflow from L.S. byte
OUT   RTS      ;OUT, done
```

Use Immediate Addressing

In general, immediate addressing is the fastest way to get a value for just about any purpose. In all instructions, immediate addressing results in a two-cycle operation as opposed to absolute addressing which burns four cycles doing the same thing.

Even if the value to be used changes occasionally, you can still get away with immediate addressing. The byte in question always follows the op-code of the relevant instruction and, therefore, has a fixed address. Simply write to it using absolute addressing when you can afford to be slow. The new value will then be there the next time you need speed.

Be careful when you calculate where to write or you'll clobber the program.

Select Branches Wisely

When you have to do a compare and branch, keep speed in mind when choosing the type of branch. For example:

```
LOOP  LDX  LOC
      CPX  #TESTVAL
      BCC  OUT
      BCS  LOOP
OUT   ---  ----        ;continue
```

This seems fine at first glance, but watch what happens if you simply reverse the order of the branches.

```
LOOP  LDX  LOC
      CPX  #TESTVAL
      BCS  LOOP
      ---  ----        ;continue
```

There's no need for the BCC since you continue execution anyway, so don't put it in. Keep a sharp eye out for this error. It seems so obvious that programmers tend to devote inadequate attention to it.

So, those are four good ones, and I think that's plenty for now. Let me once again admonish you to practice. Use the techniques. Use them even when you don't think you need them. In the long run--and for your programs there should be no such thing--you'll be glad you did.

Mr. Williams is a frequent contributor to MICRO with both articles and reviews of new products. He can be contacted at 1165 E. Edgewood Dr. 10, Ogden, Utah 84403. Please enclose S.A.S.E. with any questions.

MICRO™

Talking to Your Printer

Original Program by Dick Buchanan, Jr.

Text by Mark S. Morano

Editor's Note: The original printer program was written by Dick Buchanan for the Apple. The program was then modified and enhanced by Robert T. Tripp and Mark S. Morano.

Getting an electric typewriter to listen to a piece of fruit is not easy. If you own an Apple and a printer you probably know what we're talking about. Regardless of what micro and printer you may have you are probably familiar with the difficulty of getting your computer to talk to your printer. The size of this program should give you some idea of the scope of these problems. But don't despair --in order to use this program you are not committed to keying in the entire listing. It is an easy extract those elements that are useful to you and leave out those that are not. that are not. For example many people will never use the international style font. They would simply leave out the references to this style and the accompanying code. The same would apply to any other features you haven't any use for.

During the creation of this printer control program many interesting and frustrating problems arose. Hopefully in examining these you will find a solution applicable to your own particular problems.

To begin with each printer has its own set of printer control codes that are composed of a combination of Escape, Control and other keys, used separately and in conjunction with one another. Each printer has different control codes - why make life simple. There were a few codes that we did find in common, so these we grouped together and used for both the Epson and the Gemini. Of course this looked fine in print but we soon discovered that there was more to it than met the eye. On the Epson you must use Escape "W1" to turn Enlarge Mode on. Looking at the manual it seemed to be the same for the Gemini. We discovered that the Gemini will not accept the codes in the same form. It needed an Escape "W" CHR$(1). (Happily the Epson does accept the form needed by the Gemini). This was the kind of "obvious" bug that we ran into time and time again.

Now to make matters more complicated there is the problem of upper versus lower case letters. The printers make a distinction -the Apple doesn't. This fact was brought to our attention when the printer only printed out a line of garbled graphics. Tracing through the code we found an Escape combined with an upper case 'L' where a lower case 'l' should have been. Unfortunately, on the Apple II and II there aren't any lower case letters. As luck would have it the combination of Escape and upper case 'L' was used by the Epson to turn on the graphics mode. Undaunted we set our variable to the ASCII numeric for lower case 'l' - CHR$(108). This solution proved to be a great success, not only here but elsewhere.

Working with different printers we learned that within one machine there were certain modes that, when in operation, automatically cancelled or turned on other features. As additional styles can be obtained by using various combination of styles, it is important to know which combinations are compatible. Those styles that cannot be used together vary with each printer and should be noted so as to avoid unnecessary aggravation. For instance, with the Epson, turning on the superscript or subscript type activates the double strike style, or when using emphasized type - condensed, superscript, and subscript are not available. Usually these peculiarities do not cause any problems as everyday printing needs are not very complicated.

Another "detail" to be aware of is how your printer and computer are connected. If they are connected serially you should use PR#2 when sending an output string to the printer (ex: line 18). If you are using a parallel cable then you would use PR#3 (which is how we set the program up). The only problem you will encounter if you haven't made the right choice is that your printer won't hear your computer - rendering the program useless.

The use of output strings gives the programmer greater and easier control in matters of ports, varying differences in control codes, etc. For a further explanation of this technique read the accompanying article - String Power.

String Power

Notes Toward Generic BASIC

Concepts apply to all BASICs
Example for Apple, Commodore, CoCo and Atari

Probably the first thing everyone learns about BASIC is that:

PRINT "HELLO"

will result in the word "HELLO" being printed on the display. Then they go on to learn other ways of using the PRINT statement. Unfortunately, most of what is taught results in BASIC programs that are difficult to maintain, update, or convert to other micro BASICs. A program written in BASIC for one computer will normally **not** run on any other computer without some modification. This has been one of our greatest frustrations at MICRO. A good program submitted for micro X could be converted to run on other micros, but due to the eccentricities of BASIC, is not worth the effort. Often the differences are relatively trivial and could be avoided entirely if the programmer would use a few simple techniques to generate more *generic* code. We took the program submitted by Buchanan as a sample case and generalized it to run on several micros. One of the techniques used, **Output Strings**, can help you make your BASIC better.

Changing the Printer Port

The original program was written in standard BASIC using simple **PRINT** statements. To output a master reset to to the printer, the Applesoft BASIC statement was:

[30 *PR# 7: PRINT ESC$; CHR$(64);: PR# 0*]

which selected port 7 as the printer output port, output the characters required to reset the Gemini printer, and reset the output device to the display. Applesoft BASIC **PRINT**s to the currently selected device. Each time the device is switched between the screen (device 0 *always*) and the printer (connected as device 7 *on this system*) a **PR#** command must be issued. Every time output was directed to the printer, instead of the display, the **PR#** 7 command was issued within the print statement line. Changing the printer port would require changing every one of these print lines within the program! That is a lot of work. And, miss just one and your system will probably 'hang'. If there was some way to have the printer port defined only once in the program, then changing the printer port would only require changing one reference. There is a way. If every set of information that is destined for the printer is turned into an output string, then a subroutine that will handle the output string can be called whenever output is required. The statement:

[*OS$ = ESC$ + CHR(64)*]

defines a string variable **OS$** that contains the two characters required for a master reset. This string is output to the printer via a short subroutine:

[18 *PR# 7: PRINT OS$;: PR# 0: RETURN*]

and is called as:

[30 *OS$ = ESC$ + CHR(64): GOSUB 18*]

Every time the subroutine at line 18 is called, it selects port 7 for output, outputs the current value of the OS$ string, resets the output port to the screen, and returns. All of the statements in the original program that generated output to the printer were rewritten as output strings, using the string variable **OS$**, and calling subroutine at line 18 for the actual output. Now, if the printer is changed to port 2, then only this single line has to be changed. It is changed to:

[18 *PR# 2: PRINT OS$;: PR# 0: RETURN*]

It is obviously much easier to change the printer port when only one line needs to be changed. This makes the program a lot easier to use, maintain and update. Output strings are defined within many of the 'working' lines of the program. The basic printer control strings for the Gemini are defined in lines 9100 through 9183, plus lines 9820 through 9890.

Changing Printers

The output string technique also allows us to change printers. The original program was written only for the Gemini. To change the original code to support the Epson, or any other printer, would have required searching out every direct PR# 7 statement and changing those that were different for the Epson. Since we rewrote the program to use output strings instead, all that was required was to redefine those printer control strings that were different between the Gemini and the Epson. This is taken care of in the lines 9500 through 9583, plus lines 9000 to 9019 that allow us to choose between the Gemini and Epson at run time. Note that the Gemini and Epson people were considerate and defined many of the strings to be identical on the two printers. All of the definition strings in lines 9820 through 9850 are the same for these two popular printers.

If you wanted to modify this program for another printer, all that is required is to redefine these strings. No other program modifications should be required. If you want to key the program in for only the Gemini or the Epson, then the lines specific to the printer that you are not using may be omitted.

Changing Micros

The output string technique makes going to another micro easier. The OS$ string is defined in the exact same way, but the output subroutine is changed to fit the requirements of the new micro. In FLEX BASIC, on our FOCUS or a CoCo, the print subroutine is:

[18 *PRINT #1,OS$;: RETURN*]

On a Commodore 64 it would be:

[18 *PRINT# 4,OS$;: RETURN*]

These two BASIC implementations are quite different from the Applesoft BASIC since they specify the printer port as part of the PRINT command, rather than changing the printer port.

On an Atari it would be:

xxxxxxxx

See the micro specific listings at the end of the main program listing for the details on using one of these micros with this program.

Summary

The concept of using an output string instead of immediate printer commands has been discussed, and it has been shown that this technique can make your BASIC program more flexible. In the particular example, it made it easier to change printer port, to change type of printer, and, to change micro. This is but one of a number of techniques that can be used to make your BASIC more *generic*. Other techniques will be described in additional articles on this subject. If you have discovered other techniques, we would like to hear about them.

Listing 1

```
 1  REM  PRINTER FORMAT PROGRAM
 2  REM  ORIGINAL APPLE/GEMINI 10 VERSION
 3  REM  BY DICK BUCHANAN
 4  REM  MODIFIED BY R. M. TRIPP
 5  REM  FOR GENERAL MICROCOMPUTERS AND
 6  REM   THE EPSON PRINTER
 13  REM  MICRO - MARCH 1984 - #70
 14  REM
 15  GOSUB 9900: REM  SYSTEM INITIALIATION
 16  GOSUB 9000: GOTO 30:
    REM  PRINTER INITIALIZATION
 ****************************************
 *                                      *
 *  Microcomputer Specific Code to      *
 *  Service Input/Output Must Be        *
 *  Entered Here. See Modules at End    *
 *                                      *
 ****************************************
 30 OS$ = MR$ + MS$: GOSUB 18:
    REM  OUTPUT TO PRINTER
 50  GOSUB 20
 51 OS$ = "   A) FONT STYLE CONTROLS" + CR$:
    GOSUB 19
 52 OS$ = "   B) FONT PITCH CONTROLS" + CR$:
    GOSUB 19
 53 OS$ = "   C) SPECIAL PRINT MODES" + CR$:
    GOSUB 19
 54 OS$ = "   D) SPECIAL PRINT EFFECTS" + CR$:
    GOSUB 19
 55 OS$ = "   E) LINE FEED CONTROLS" + CR$: GOSUB 19
 56 OS$ = "   F) FORM FEED CONTROLS" + CR$: GOSUB 19
 57 OS$ = "   G) VERTICAL TABS" + CR$: GOSUB 19
 58 OS$ = "   H) HORIZONTAL CONTROLS" + CR$:
    GOSUB 19
 59 OS$ = "   I) INITIALIZE PRINTER RESET" + CR$:
    GOSUB 19
 60 OS$ = "   J) PRINT PRESENT PARAMETERS" + CR$:
    GOSUB 19
 61 OS$ = "   K)
    DISPLAY PRESENT PARAMETERS" + CR$ + CR$:
    GOSUB 19
 63 OS$ = "   X) TO EXIT" + CR$: GOSUB 19
 65 OS$ = CR$: GOSUB 19
 70 OS$ = SN$: GOSUB 19: GOSUB 21
 75  IF N$ = "" THEN 70
 80  IF N$ = "X" THEN  END
 90 N =  ASC (N$) -  ASC ("@"):
    IF N < 1 OR N > 11 THEN 70
100  ON N GOSUB 1000,2000,3000,4000,5000,6000,7000,
    8000,30,300,400
110  GOTO 50
200  REM   PRESENT VALUES
205  GOSUB 20
210 OS$ = AA$ + " " + BS$ + CQ$ + B3$ + CQ$ + A2$ +
    PT$ + CR$
211 OS$ = OS$ + CS$ + CQ$ + C2$ + CQ$ + DS$ + CR$
212 OS$ = OS$ + RS$ + RC$ + CQ$ + LS$ + LC$ + CR$
213 OS$ = OS$ + D2$ + CQ$ + LF$ + " " + LT$ + "/" +
    LB$ +  CHR$ (34) + CR$
214 OS$ = OS$ + PL$ + FS$ + CQ$ + PN$ + F1$ +  CHR$
    (34) + CQ$
215 OS$ = OS$ + HL$ + F2$ + CQ$ + BL$ + F3$ + CR$
220  RETURN
299  REM   DISPLAY PRESENT VALUES
300 CQ$ = ", ": GOSUB 210: GOTO 18:
    REM  OUTPUT TO PRINTER
400 CQ$ = CR$: GOSUB 205: GOSUB 19: GOTO 21:
    REM  OUTPUT TO DISPLAY
1000  REM  FONT STYLE
1005  GOSUB 20
1010 OS$ = " 1) SELECT STANDARD ASCII" + CR$
1011 OS$ = OS$ + " 2) SELECT ITALIC" + CR$
1012 OS$ = OS$ + " 3) SELECT INTERNATIONAL" + CR$:
    GOSUB 19
1040 OS$ = SN$: GOSUB 19: GOSUB 21:
    IF N$ = "" THEN  RETURN
1045  IF N$ < "1" OR N$ > "3" THEN 1040
1050  IF N$ = "1" THEN OS$ = IX$: GOSUB 18:
    AA$ = "STANDARD ASCII": RETURN
1060  IF N$ = "2" THEN OS$ = IN$: GOSUB 18:
    AA$ = "ITALIC PRINT": RETURN
1070  IF N$ <  > "3" THEN 1080
1071  GOSUB 20:OS$ = "INTERNATIONAL FONTS" + CR$:
    GOSUB 19
1072  FOR I = 0 TO AX:
    OS$ =  CHR$ (I + 48) + " = " + AX$(I) + CR$
1073  GOSUB 19: NEXT I
1075 OS$ = SN$: GOSUB 19: GOSUB 21:
    IF N$ = "" THEN  RETURN
1080  IF N$ < "0" OR N$ >  CHR$ (AX + 48) THEN 1075
1090 AA$ = AX$( VAL (N$))
1100 OS$ = SI$ +  CHR$ ( VAL (N$)): GOSUB 18:
    RETURN
2000  REM  FONT PITCH
2005  GOSUB 20
2010 OS$ = " 1) SET PICA STANDARD  (" + PX$ + ")
    " + CR$
2011 OS$ = OS$ + " 2)
    SET ELITE STANDARD (" + EX$ + ")" + CR$
2012 OS$ = OS$ + " 3)
    SET CONDENSED      (" + CX$ + ")" + CR$
2013 OS$ = OS$ + " 4)
    SET ENLARGED       (" + EN$ + ")" + CR$
2014 OS$ = OS$ + " 5) CANCEL ENLARGED MODE" + CR$:
    GOSUB 19
2050 OS$ = SN$: GOSUB 19: GOSUB 21:
    IF N$ = "" THEN  RETURN
2060  IF N$ < "1" OR N$ > "5" THEN 2050
2065  ON  VAL (N$) GOTO 2110,2120,2130,2140,2150
2110 OS$ = PF$ + ES$ +  CHR$ (81) + PC$: GOSUB 18:
    PT$ = PX$
```

```
2111 RC$ = PC$:BS$ = "PICA": RETURN
2120 OS$ = EF$ + ES$ +  CHR$ (81) + EC$: GOSUB 18:
     PT$ = EX$
2121 RC$ = EC$:BS$ = "ELITE": RETURN
2130 OS$ = CF$ + ES$ +  CHR$ (81) + CC$: GOSUB 18:
     PT$ = CX$
2131 RC$ = CC$:BS$ = "CONDENSED": RETURN
2140 OS$ = EM$: GOSUB 18:B3$ = "ENLARGED": RETURN
2150 OS$ = EY$: GOSUB 18:B3$ = "NON-ENLARGED":
     RETURN
3000  REM  SPECIAL PRINT
3005  GOSUB 20
3010 OS$ = " 1) DOUBLE STRIKE PRINT" + CR$
3011 OS$ = OS$ + " 2) CANCEL DOUBLE STRIKE" + CR$
3012 OS$ = OS$ + " 3) EMPHASIZED MODE" + CR$
3013 OS$ = OS$ + " 4)
     CANCEL EMPHASIZED MODE" + CR$ + CR$
3050  GOSUB 19
3060 OS$ = SN$: GOSUB 19: GOSUB 21:
     IF N$ = "" THEN  RETURN
3070  IF N$ < "1" OR N$ > "4" THEN 3060
3090  ON  VAL (N$) GOTO 3110,3120,3130,3140
3110 CS$ = "DOUBLE STRIKE":OS$ = SD$: GOSUB 18:
     GOTO 3005
3120 CS$ = "NON-DOUBLE STRIKE":OS$ = CD$: GOSUB 18:
     GOTO 3005
3130 C2$ = "EMPHASIZED":OS$ = SE$: GOSUB 18:
     GOTO 3005
3140 C2$ = "NON-EMPHASIZED":OS$ = CE$: GOSUB 18:
     GOTO 3005
4000  REM  SPECIAL EFFECTS
4005  GOSUB 20
4010 OS$ = " 1) UNDERLINE CHARACTERS" + CR$
4011 OS$ = OS$ + " 2) CANCEL UNDERLINE" + CR$
4012 OS$ = OS$ + " 3) SUPERSCRIPT MODE" + CR$
4013 OS$ = OS$ + " 4) SUBSCRIPT MODE" + CR$
4014 OS$ = OS$ + " 5)
     CANCEL SUPER/SUBSCRIPT MODE" + CR$
4015 OS$ = OS$ + " 6) UNI-DIRECTIONAL MODE" + CR$
4016 OS$ = OS$ + " 7)
     BI-DIRECTIONAL MODE" + CR$ + CR$
4080  GOSUB 19
4090 OS$ = SN$: GOSUB 19: GOSUB 21:
     IF N$ = "" THEN  RETURN
4100  IF N$ < "1" OR N$ > "7" THEN 4090
4106 N =  VAL (N$):
     ON N GOTO 4110,4120,4130,4140,4150,4160,4170
4110 OS$ = UN$: GOSUB 18: GOTO 4005
4120 OS$ = UF$: GOSUB 18: GOTO 4005
4130 DS$ = "SUPERSCRIPT MODE":OS$ = SS$: GOSUB 18:
     GOTO 4005
4140 DS$ = "SUBSCRIPT MODE":OS$ = SB$: GOSUB 18:
     GOTO 4005
4150 DS$ = "NORMAL MODE":OS$ = SF$: GOSUB 18:
     GOTO 4005
4160 D2$ = "UNI-DIRECTIONAL MODE":OS$ = UD$:
     GOSUB 18: GOTO 4005
4170 D2$ = "BI-DIRECTIONAL MODE":OS$ = BD$:
     GOSUB 18: GOTO 4005
5000  REM  LINE FEED CONTROLS
5005  GOSUB 20
5010 OS$ = " 1) SET LF TO 9/72 (1/8) INCH" + CR$
5011 OS$ = OS$ + " 2) SET LF TO 7/72 INCH" + CR$
5012 OS$ = OS$ + " 3) SET LF TO 12/72 (1/6)
     INCH" + CR$
5013 OS$ = OS$ + " 4)
     SET LF TO N/72 INCH (N=1 TO 127)" + CR$
5014 OS$ = OS$ + " 5)
     SET LF TO N/" + HM$ + " INCH (N=1 TO 127)
     " + CR$ + CR$
5060  GOSUB 19
5070 OS$ = SN$: GOSUB 19: GOSUB 21:
     IF N$ = "" THEN  RETURN
5080  IF N$ < "1" OR N$ > "5" THEN 5070
5100 N =  VAL (N$):
     ON N GOTO 5110,5120,5130,5140,5140
5110 LT$ = "1":LB$ = "8":OS$ = L0$: GOTO 18
5120 LT$ = "7":LB$ = "72":OS$ = L1$: GOTO 18
5130 LT$ = "1":LB$ = "6":OS$ = L2$: GOTO 18
5140  INPUT "ENTER N (1-127): ";P$
5142  IF  VAL (P$) < 1 OR  VAL (P$)
     > 127 THEN  GOTO 5140
5145  IF  VAL (P$) > 0 OR  VAL (P$)
     < 128 THEN LT$ = P$: ON N-3 GOTO 5160,5170
5160 LB$ = "72":OS$ = L3$ +  CHR$ ( VAL (P$)):
     GOTO 18
5170 LB$ = HM$:OS$ = L4$ +  CHR$ ( VAL (P$)):
     GOTO 18
6000  REM  FORM FEED CONTROLS
6005  GOSUB 20
6010 OS$ = " 1) SET LINES PER PAGE (1-127)" + CR$
6011 OS$ = OS$ + " 2)
     SET PAGE LENGTH IN INCHES (1-32)" + CR$
6012 OS$ = OS$ + " 3) SET HEADER LINE" + CR$
6013 OS$ = OS$ + "    (FIRST LINE PRINTED, 1-16)
     " + CR$
6014 OS$ = OS$ + " 4)
     SET MAXIMUM LINES FROM BOTTOM OF" + CR$
6015 OS$ = OS$ + "    THE PAGE (1-127)" + CR$
6016 OS$ = OS$ + " 5)
     CANCEL LINES FROM BOTTOM SETTING" + CR$ + CR$
6060  GOSUB 19
6070 OS$ = SN$: GOSUB 19: GOSUB 21:
     IF N$ = "" THEN  RETURN
6080  IF N$ < "1" OR N$ > "5" THEN 6070
6090 N =  VAL (N$):
     ON N GOTO 6110,6120,6130,6140,6150
6110  INPUT "ENTER LLP (1-127): ";P$
6111  IF  VAL (P$) < 1 OR  VAL (P$) > 127 THEN 6110
6115 FS$ = P$:OS$ = FL$ +  CHR$ ( VAL (P$)):
     GOSUB 18: GOTO 6005
6120  INPUT "ENTER PL (1-32): ";P$:
     IF  VAL (P$) < 1 OR VAL (P$) > 32 THEN 6120
6125 F1$ = P$:OS$ = FI$ +  CHR$ ( VAL (P$)):
     GOSUB 18: GOTO 6005
6130  INPUT "ENTER HL (1-16): ";P$:
     IF  VAL (P$) < 1 OR VAL (P$) > 16 THEN 6130
6135 F2$ = P$:OS$ = HD$ +  CHR$ ( VAL (P$)):
     GOSUB 18: GOTO 6005
6140  INPUT "ENTER LFB (1-127): ";P$
6141  IF VAL (P$) < 1 OR VAL (P$) > 127 THEN 6140
6145 F3$ = P$:OS$ = SO$ +  CHR$ ( VAL (P$)):
     GOSUB 18: GOTO 6005
6150 OS$ = SX$: GOSUB 18: GOTO 6005
7000  REM  VERTICAL TABS
7005  GOSUB 20
7010 OS$ = " 1) ADVANCE TO NEXT TAB" + CR$
```

```
7011 OS$ = OS$ + "    (6,12,18, ... ,60 STANDARD)
     " + CR$
7012 OS$ = OS$ + " 2)
     SET NEW VERTICAL TAB POSITIONS" + CR$
7013 OS$ = OS$ + "    (MAXIMUM OF 20)" + CR$
7014 OS$ = OS$ + CR$ + CR$
7040  GOSUB 19
7050 OS$ = SN$: GOSUB 19: GOSUB 21:
     IF N$ = "" THEN  RETURN
7060  IF N$ = "1" THEN OS$ = VT$: GOSUB 18:
     GOTO 7005
7065 J = 0: IF N$ < > "2" THEN 7050
7070  INPUT "ENTER TAB: ";T$(J + 1):J = J + 1
7075  INPUT "SET NEXT TAB (Y/N): ";P$:
     IF  LEFT$ (P$,1) = "Y" THEN 7070
7080 OS$ = SV$: OS$ = TS$: FOR I = 1 TO J:
     OS$ = OS$ + CHR$(VAL (T$(I))):NEXT:GOSUB 18
7085  FOR I = 1 TO J:
     OS$ = OS$ +  CHR$ ( VAL (T$(I))): NEXT
7090 OS$ = OS$ +  CHR$ (0): GOSUB 18: GOTO 7005
8000  REM  HORIZONTAL CONTROLS
8005  GOSUB 20
8010 OS$ = " 1) SEND CARRIAGE RETURN" + CR$
8011 OS$ = OS$ + " 2) SET LEFT MARGIN" + CR$
8012 OS$ = OS$ + " 3) SET RIGHT MARGIN" + CR$
8013 OS$ = OS$ + " 4)
     MOVE TO NEXT HORIZONTAL TAB" + CR$
8014 OS$ = OS$ + "    (10,20,30,
     ...SET STANDARD" + CR$
8015 OS$ = OS$ + CR$ + " 5)
     SET NEW TAB POSITIONS" + CR$ + CR$
8070  GOSUB 19
8080 OS$ = SN$: GOSUB 19: GOSUB 21:
     IF N$ = "" THEN  RETURN
8090  IF N$ < "1" OR N$ > "5" THEN 8080
8100  ON  VAL (N$) GOTO 8110,8120,8130,8140,8150
8110 OS$ = CR$: GOSUB 18: GOTO 8005
8120  INPUT "ENTER LEFT MARGIN: ";P$:LC$ = P$:
     OS$ = SL$ +  CHR$ ( VAL (P$))
8125  GOSUB 18: GOTO 8005
8130  INPUT "ENTER RIGHT MARGIN: ";P$:RC$ = P$:
     OS$ = SR$ +  CHR$ ( VAL (P$))
8135  GOSUB 18: GOTO 8005
8140 OS$ =  CHR$ (9): GOSUB 18: GOTO 8005
8150 J = 0
8155  INPUT "ENTER TAB: ";T$(J + 1):J = J + 1:
8160  INPUT "  SET NEXT TAB (Y/N): ";P$:
     IF  LEFT$ (P$,1) = "Y" THEN 8155
8165 OS$ = TS$: FOR I = 1 TO J:
     OS$ = OS$ +  CHR$ ( VAL (T$(I))): NEXT
8170 OS$ = OS$ +  CHR$ (0): GOSUB 18: GOTO 8005
8999  REM  INITIALIZATION
9000  REM  APPLE VERSION
9010  GOSUB 20: INPUT "GEMINI OR EPSON [G/E]: ";TY$
9011  IF TY$ = "G" THEN TY = 1: GOTO 9100
9012  IF TY$ = "E" THEN TY = 2: GOTO 9500
9019  GOTO 9010
```

GEMINI

```
9100  REM   GEMINI CHARACTER SET STRINGS
9101  DIM AX$(8)
9102 AX$(0) = "USA"
9103 AX$(1) = "ENGLAND"
9104 AX$(2) = "GERMANY"
9105 AX$(3) = "DENMARK"
9106 AX$(4) = "FRANCE"
9107 AX$(5) = "SWEDEN"
9108 AX$(6) = "ITALY"
9109 AX$(7) = "SPAIN"
9110 AX = 7: REM  NUMBER OF COUNTRIES
9120  REM   GEMINI COMMAND STRINGS
9131 SI$ = ES$ + "7"
9140 PF$ = ES$ + "B" +  CHR$ (1): REM  PICA MODE
9150 EF$ = ES$ + "B" +  CHR$ (2): REM  ELITE MODE
9160 CF$ = ES$ + "B" +  CHR$ (3):
     REM  CONDENSED MODE
9170 SS$ = ES$ + "S" +  CHR$ (0):
     REM  SUPERSCRIPT MODE
9171 SB$ = ES$ + "S" +  CHR$ (1):
     REM  SUBSCRIPT MODE
9180 SL$ = ES$ + "M": REM  SET LEFT MARGIN
9181 SV$ = ES$ + "P": REM  SET VERTICAL TAB
9182 SR$ = ES$ + "Q": REM  RIGHT MARGIN
9183 HM$ = "144": REM  HIGH DENSITY LINES
9300  GOTO 9800
```

SPECIFIC CODE

```
9500  REM   EPSON CHARACTER SET STRINGS
9501  DIM AX$(9)
9502 AX$(0) = "USA"
9503 AX$(1) = "FRANCE"
9504 AX$(2) = "GERMANY"
9505 AX$(3) = "ENGLAND"
9506 AX$(4) = "DENMARK"
9507 AX$(5) = "SWEDEN"
9508 AX$(6) = "ITALY"
9509 AX$(7) = "SPAIN"
9510 AX$(8) = "JAPAN"
9511 AX = 8: REM  NUMBER OF COUNTRIES
9520  REM  EPSON COMMAND STRINGS
9531 SI$ = ES$ + "R": REM  SET INTERNATIONAL
9540 PF$ = ES$ + "P": REM  PICA MODE
9550 EF$ = ES$ + "M": REM  ELITE MODE
9560 CF$ =  CHR$ (15): REM  CONDENSED MODE
9570 SS$ = ES$ + "S" +  CHR$ (1):
     REM  SUPERSCRIPT MODE
9571 SB$ = ES$ + "S" +  CHR$ (0):
     REM  SUBSCRIPT MODE
9580 SL$ = ES$ +  CHR$ (108): REM  SET LEFT MARGIN
9581 SV$ = ES$ + "B": REM  SET VERTICAL TAB
9582 HM$ = "216": REM  HIGH DENSITY LINE MODE
9583 SR$ = ES$ + "Q" : REM  RIGHT MARGIN
```

EPSON SPECIFIC CODE

```
9800  REM   STANDARD STRINGS
9801 AA$ = "STANDARD ASCII":A2$ = "FONT PITCH IS "
9802 BS$ = "PICA":PT$ = "10 CPI/80 CPL"
9803 B3$ = "NON-ENLARGED":CS$ = "NON-DOUBLE STRIKE"
9804 C2$ = "NON-EMPHASIZED":RC$ = "80":LC$ = "1"
9805 RS$ = "RIGHT MARGIN SET TO ":
     DS$ = "NORMAL MODE"
9806 LS$ = "LEFT MARGIN SET TO ":LT$ = "1":
     LB$ = "6"
9807 D2$ = "BI-DIRECTIONAL MODE":
     LF$ = "LINE FEED IS"
9808 PL$ = "LLP= ";PN$ = "PL= ":HL$ = "HL= ":
     BL$ = "LFB= "
9809 FS$ = "66":F1$ = "11":F2$ = "1":F3$ = "0"
9819  REM
9820  REM  COMMON COMMAND STRINGS - GEMINI AND
     EPSON
9821 MR$ = ES$ +  CHR$ (64): REM  MASTER RESET
9822 SR$ = ES$ + "Q" +  CHR$ (80):
     REM  SET RIGHT MARGIN 80
```

```
9823 IN$ = ES$ + "4": REM  ITALIC ON
9824 IX$ = ES$ + "5": REM  ITALIC OFF
9825 EM$ = ES$ + "W" + CHR$(1) :
     REM  ENLARGED MODE ON
9826 EY$ = ES$ + "W" + CHR$(Ø) :
     REM  ENLARGED MODE OFF
9827 UN$ = ES$ + CHR$(-1) : REM  UNDERLINE MODE ON
9828 UF$ = ES$ + CHR$(-Ø) : REM  UNDERLINE MODE OFF
9829 NM$ = ES$ + "T": REM  TURN SCRIPT MODE OFF
983Ø UD$ = ES$ + "U" + CHR$(1)  :
     REM  UNI-DIRECTIONAL MODE
9831 BD$ = ES$ + "U" + CHR$(Ø)  :
     REM  BI-DIRECTIONAL MODE
9832 LØ$ = ES$ + "Ø": REM  1/8 INCH LINE SPACE
9833 L1$ = ES$ + "1": REM  7/72 INCH LINE SPACE
9834 L2$ = ES$ + "2": REM  1/6 INCH LINE SPACE
9835 L3$ = ES$ + "A": REM  X/72 INCH LINE SPACE
9836 L4$ = ES$ + "3":
     REM  X/144 GEMINI, X/216 EPSON
9837 FL$ = ES$ + "C": REM  FORM LINES
9838 FI$ = ES$ + "C" +  CHR$ (Ø):
     REM  FORM LENGTH IN INCHES
9839 SO$ = ES$ + "N": REM  SKIP OVER PERFORATION
984Ø SX$ = ES$ + "O": REM  TURN SKIP OVER OFF
9841 VT$ =  CHR$ (11): REM  EXECUTE VERTICAL TAB
9842 SR$ = ES$ + "Q": REM  SET RIGHT MARGIN
9843 TS$ = ES$ + "D": REM  HORIZONTAL TAB SET
9844 SE$ = ES$ + "E": REM  SET EMPHASIZED MODE
9845 CE$ = ES$ + "F": REM  CLEAR EMPHASIZED MODE
9846 SD$ = ES$ + "G": REM  SET DOUBLE STRIKE MODE
9847 CD$ = ES$ + "H": REM  CLEAR DOUBLE STRIKE MODE
985Ø HD$ = ES$ + "R": REM  HEADER LINE
989Ø  RETURN
9899  REM
99ØØ  REM  SYSTEM SPECIFIC STUFF
99Ø5 ES$ =  CHR$ (27): REM   ESCAPE CODE
99Ø6 CR$ =  CHR$ (13):
     REM  CARRIAGE RETURN/LINEFEED
99Ø7 SN$ = CR$ + "SELECT:": REM  SELECT MESSAGE
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$                                      $
$  Printer Initialization Code, If     $
$  Required, Must Be Entered Here.     $
$                                      $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$                                       $
$  The following statements should be   $
$  altered according to your printer.   $
$  As we had an 8 inch printer we set   $
$  1Ø CPI/8Ø CPL (line 9912-9917)       $
$                                       $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
9911 PX$ = "1Ø CPI/8Ø CPL"
9912 PC = 8Ø: REM  CHARACTERS PER LINE
9913 EX$ = "12 CPI/96 CPL"
9914 EC = 96: REM  CHARACTERS PER LINE
9915 CX$ = "17 CPI/136 CPL"
9916 CC = 136: REM  CHARACTERS PER LINE
9917 EN$ = "5, 6, 8.5 CPI"
992Ø  RETURN
```

EPSON & GEMINI

modules

Subroutines for Apple II

```
18 PR# 7:PRINT OS$;:PR# Ø:RETURN: REM OUTPUT TO PRINTER
19 PRINT OS$;:RETURN: REM OUTPUT TO DISPLAY
2Ø HOME:RETURN:  REM HOME COMMAND
21 INPUT N$:RETURN
```

Subroutines for FLEX used on Color Computer

```
18 PRINT #Ø,OS$;:RETURN: REM OUTPUT TO PRINTER
19 PRINT OS$;:RETURN: REM OUTPUT TO DISPLAY
2Ø PRINT CHR$(12);:RETURN:  REM CLEAR SCREEN
21 INPUT #Ø,N$:RETURN

991Ø OPEN "Ø.PRINT.SYS" AS Ø: REM OPEN PRINTER DEVICE
```

Subroutines for FLEX used on FOCUS

```
18 PRINT #Ø,OS$;:RETURN: REM OUTPUT TO PRINTER
19 PRINT OS$;:RETURN: REM OUTPUT TO DISPLAY
2Ø PRINT CHR$(11);CHR$(24);:RETURN :REM CLEAR SCREEN
21 INPUT #Ø,N$:RETURN

991Ø OPEN "Ø.PARALLEL.CMD" AS Ø
```

Subroutines for Commodore 64/VIC-2Ø

```
18 PRINT# 1,OS$;:RETURN: REM OUTPUT TO PRINTER
19 PRINT OS$;:RETURN: REM OUTPUT TO DISPLAY
2Ø PRINT (CLEAR);:RETURN :REM CLEAR SCREEN
21 INPUT N$:RETURN

991Ø CLOSE 1:OPEN 1,4: REM OPEN PRINTER PORT
```

Subroutine for the Atari

Unfortunately this program could not be adapted to the Atari. The problem arose in the output strings that had to be concatenated. Given how laboriously Atari handles concatenation and how often it would have to be used in this program, conversion was made virtually impossible.

MICRO

HI-RES SCREEN DUMP for the EPSON MX-80

by Robert D. Walker

A machine language subroutine for dumping high resolution Apple II graphics to the Epson MX-80 printer which allows choice of screen dump size.

Requirements:
Apple II with 48K
Epson MX-80 equipped with Graftrax
8 bit parallel interface

In the February 1983 issue of MICRO I published a short article which included an Apple Pascal program for printing the Apple II HiRes (abbreviation for high resolution) graphics with the Epson MX-80 equipped with the Graftrax option. For those with a parallel interface capable of sending 8 bits, this program worked flawlessly, but slowly. Let's face it, the Apple p-code interpreter is generally faster than BASIC, but it is not exceptionally fast. In addition, one must realize that the entire HiRes screen contains 53,760 pixels. Each pixel must be processed individually, this accounting for the slow execution of this program.

In this article I have included a fast 768 byte machine language subroutine which dumps the HiRes screen to the Epson MX-80. In addition to the usual dot-for-dot format (see figure 1), I have included an optional format for creating an expanded printout (see figure 2). Careful examination of figure 2 will reveal that each screen pixel is printed as a two by two dot matrix.

To demonstrate the method of calling this machine language subroutine from your own BASIC program, I have included a useful

Figure 1 Normal Size Screen Dump

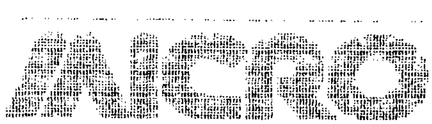


Figure 2 Expanded Size Screen Dump

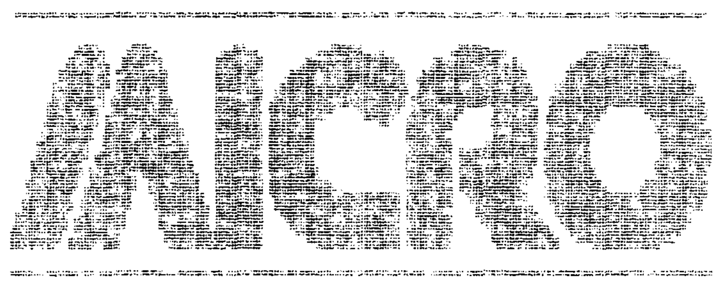


Table 1
Screen Dump Memory Locations

Location Hex	Location Dec	Explanation	Default Value
$9300	37632	call this location to dump the screen normal size	
$9303	37635	call this location to dump the screen expanded size	
$9306	37638	screen page, for HiRes page 1 use $20 (32), for page 2 use $40 (63)	32
$9307	37639	left column of screen area to be dumped DIV 7	0
$9308	37640	right column of screen area to be dumped DIV 7	39
$9309	37641	top row of screen area to be dumped DIV 8	0
$930A	37642	bottom row of screen area to be dumped DIV 8	23
$930B	37643	number of spaces in left margin of normal size dump	16
$930C	37644	number of spaces in left margin of expanded size dump	10
$930D	37645	byte exclusive-ored with image, 0 = normal image, 255 = reversed image.	0

Listing 2

```
10  REM ****************************************
11  REM *                                      *
12  REM * PROGRAM:  APPLE II -> EPSON MX-80    *
13  REM *           HIRES SCREEN DUMP          *
14  REM *                                      *
15  REM * AUTHOR:  ROBERT D. WALKER            *
16  REM *          MARIETTA, GA                *
17  REM *                                      *
19  REM *                                      *
20  REM ****************************************
21  REM ** INITIALIZE **
100  HIMEM: 37631
110 D$ =  CHR$ (4)
120  PRINT D$;"BLOAD OBJ.DUMP,A$9300"
130 SC = 32: REM     SCREEN PAGE
140 LC = 0: REM     LEFT SCREEN COLUMN
150 RC = 279: REM    RIGHT SCREEN COLUMN
160 TR = 0: REM     TOP SCREEN ROW
170 BR = 191: REM    BOTTOM SCREEN ROW
180 L1 = 16: REM     LEFT MARGIN (NORMAL SIZE DUMP)
190 L2 = 10: REM     LEFT MARGIN (EXPANDED DUMP)
200 IM = 0: REM      IMAGE (0=NORMAL, 255=INVERSE)
210  REM ** GET PICTURE **
220  HOME
230  VTAB (3): HTAB (8): PRINT "APPLE II -> EPSON MX-80"
240  HTAB (11): PRINT "HIRES SCREEN DUMP"
250  VTAB (10): INPUT "FILE NAME- ";F$
260  IF  LEN (F$) = 0 THEN  END
270  HOME : FLASH : PRINT "LOADING ";F$: NORMAL
280  PRINT D$;"BLOAD ";F$;",A$2000"
290  POKE  - 16297,0
300  POKE  - 16304,0
310  POKE  - 16302,0
320  FOR I = 1 TO 1000: NEXT
330  POKE  - 16301,0
340  VTAB (23): PRINT "HIT SPACE TO CONTINUE...";
350  GET A$
360  REM ** SELECT SIZE AND PARAMETERS **
370  TEXT : HOME
380  PRINT "N(ORMAL SIZE DUMP"
390  PRINT "E(XPANDED DUMP"
400  PRINT : GET A$
410  IF A$ <  > "N" AND A$ <  > "E" GOTO 370
420  IF A$ = "N" THEN F2 = 1: GOTO 440
430 F2 = 2
440  HOME
450  PRINT "SCREEN DUMP PARAMETERS:"
460  PRINT
470  PRINT "  1.  SCREEN PAGE";: HTAB (35): PRINT SC
480  PRINT "  2.  LEFT SCREEN COLUMN";: HTAB (35):
     PRINT LC
490  PRINT "  3.  RIGHT SCREEN COLUMN";: HTAB (35):
     PRINT RC
500  PRINT "  4.  TOP SCREEN ROW";: HTAB (35): PRINT TR
510  PRINT "  5.  BOTTOM SCREEN ROW";: HTAB (35):
     PRINT BR
520  PRINT "  6.  LEFT MARGIN";: HTAB (35)
530  IF F2 = 1 THEN  PRINT L1: GOTO 550
540  PRINT L2
550  PRINT "  7.  IMAGE";: HTAB (35): PRINT IM
560  VTAB (15): PRINT "ENTER PARAMETER NUMBER TO MAKE
     CHANGES."
570  PRINT "WHEN NO MORE, HIT RETURN."
580  VTAB (20): GET A$
590 A =  ASC (A$) - 48
600  IF A =  - 35 GOTO 720
610  ON A GOTO 630,640,650,660,670,680,710
620  GOTO 440
630  INPUT "SCREEN PAGE- ";SC: GOTO 440
640  INPUT "LEFT SCREEN COLUMN- ";LC: GOTO 440
650  INPUT "RIGHT SCREEN COLUMN- ";RC: GOTO 440
660  INPUT "TOP SCREEN ROW- ";TR: GOTO 440
670  INPUT "BOTTOM SCREEN ROW- ";BR: GOTO 440
680  INPUT "LEFT MARGIN- ";L
690  IF F2 = 1 THEN L1 = L: GOTO 440
700 L2 = L: GOTO 440
710  INPUT "IMAGE- ";IM: GOTO 440
720  REM ** SEND PARAMETERS TO MACHINE LANGUAGE ROUTINE
730  POKE 37638,SC
740  POKE 37639, INT (LC / 7)
750  POKE 37640, INT (RC / 7)
760  POKE 37641, INT (TR / 8)
770  POKE 37642, INT (BR / 8)
780  POKE 37643,L1
790  POKE 37644,L2
800  POKE 37645,IM
810  REM ** DUMP IMAGE TO PRINTER **
820  HOME : VTAB (10): FLASH : PRINT "DUMPING IMAGE.":
     NORMAL
830  PRINT "HIT ESC TO STOP AT ANY TIME..."
840  IF F2 = 1 THEN  CALL 37632: GOTO 220
850  CALL 37635: GOTO 220
```

Listing 1

```
     ;*************************************
     ;*                                   *
     ;*    APPLE II -> EPSON MX-80        *
     ;*   HIRES SCREEN DUMP ROUTINES      *
     ;*                                   *
     ;*       ROBERT D. WALKER            *
     ;*         MARIETTA, GA              *
     ;*                                   *
     ;*************************************
     ;
     ;***********************************
     ;* SUBROUTINE WRITTEN FOR THE      *
     ;* EPSON MX-80 WITH GRAFTRAX OPTION *
     ;*                                 *
     ;* HIMEM SHOULD BE SET LOWER THAN  *
     ;* 37632 ($9300) FOR A 48K SYSTEM  *
     ;***********************************
     ;
     ; ZERO PAGE USAGE
00FC BLKPT    EQU $00FC       ; POINTER TO BLOCK
00FE TBLKPT   EQU $00FE       ; TEMP POINTER TO BLOCK
     ;
     ; MISC CONSTANTS AND LOCATIONS
009B ESC      EQU $009B       ; INTERRUPT KEY
C000 KEYDATA  EQU $C000       ; KEYBOARD DATA LOCATION
C010 KEYSTRB  EQU $C010       ; KEYBOARD STROBE LOCATION
     ;
     ; PRINTER SLOT #1
C090 PROUT    EQU $C090       ; PRINTER OUTPUT LOCATION
C1C1 PRWAIT   EQU $C1C1       ; CHECK IF PRINTER READY LOC
     ;
```

```
9300                    ORG $9300
                  ;
                  ; ROUTINE ENTRY POINTS
9300 4C 22 93           JMP DUMP
9303 4C 98 93           JMP DUMPE
                  ;
                  ; PARAMETERS
9306 20           SCRNPG   BYT $20
9307 00           LCOL     BYT $00
9308 27           RCOL     BYT $27
9309 00           TROW     BYT $00
930A 17           BROW     BYT $17
930B 10           LMARG    BYT $10
930C 0A           LMARGE   BYT $0A
930D 00           IMAGE    BYT $00
                  ;
                  ; TEMP STORAGE
930E 00           ROW      BYT 0
930F 00           COL      BYT 0
9310 00           BLKROW   BYT 0
9311 00           BLKCOL   BYT 0
9312 00           EXBYTE   BYT 0
9313 00           TIME     BYT 0
9314 00           MULT1    BYT 0
9315 00           MULT2    BYT 0
9316 00 00        PROD     BYT 0,0
9318 00 00        DOTS     BYT 0,0
931A 00 00 00     BLKTAB   BYT 0,0,0,0,0,0,0,0
                  ;
                  ;**********************************
                  ;* DUMP THE HIRES SCREEN, DOT FOR *
                  ;* DOT -- NORMAL SIZE             *
                  ;**********************************
                  ;
9322 A9 08        DUMP     LDA #$08        ; LINE SPACING
9324 20 77 95              JSR LINESPC     ; - 8 DOTS
9327 AD 09 93              LDA TROW        ; START AT TOP
932A 8D 0E 93              STA ROW
                  ;
                  ; PRINT ONE ROW OF BLOCK --
                  ; I.E. 8 ROWS OF DOTS
932D 20 64 95     PROW     JSR CHKKEY      ; INTERRUPT ?
9330 AD 0B 93              LDA LMARG       ; LEFT MARGIN
9333 20 89 95              JSR TAB
9336 AD 07 93              LDA LCOL        ; START AT LEFT
9339 8D 0F 93              STA COL
933C 38                    SEC             ; COMPUTER # DOTS
933D AD 08 93              LDA RCOL        ; IN ONE ROW
9340 ED 07 93              SBC LCOL
9343 8D 14 93              STA MULT1
9346 EE 14 93              INC MULT1
9349 A9 07                 LDA #$07
934B 8D 15 93              STA MULT2
934E 20 44 95              JSR MULTPLY
9351 20 B6 95              JSR PDOTS       ; TELL PRINTER DOTS IN ROW
9354 AD 16 93              LDA PROD
9357 20 9F 95              JSR PRCOUT
935A AD 17 93              LDA PROD+1
935D 20 9F 95              JSR PRCOUT
9360 20 C0 94     PBLK     JSR CALCBLK     ; CALC ADDRESS
9363 20 A3 94              JSR ROTBLK      ; ROTATE IMAGE
9366 A2 00                 LDX #$00
9368 BD 1A 93     PCOL     LDA BLKTAB,X    ; GET BYTE
```


Subscribers - Check Your Labels !

As part of our recent reorganization, we have changed subscription fulfillment houses.

As programmers, you can appreciate that any change such as this, no matter how carefully done, is likely to cause some errors during the transition period. This is especially true with the foreign subscriptions. While the new system uses a better method of generating country information, we had to make some guesses in converting from the existing labels.

Please examine your subscription label on this issue of MICRO. If you find any problems - wrong address, incorrect spelling, incorrect zip code, etc. - please notify us immediately.

MICRO Circulation
P.O. Box 6502
Chelmsford, MA 01824

Applesoft program for loading and printing HiRes pictures.

The Machine Language Subroutine

Listing 1 shows the assembly listing of the screen dump subroutine. In this listing the subroutine was assembled to reside in memory locations $9300 through $95FF, just below DOS for a 48K system. Table 1 shows the important memory locations for calling this subroutine.

As shown in Table 1, the area of the screen to be printed is determined by four parameters: left ($9307), right ($9308), top ($9309), bottom ($930A). These parameters require special consideration.

The left and right parameters are each divided by 7. Assume,for example, that the entire screen is to be printed. The far left column would be 0, while the far right column would be 279. Dividing both of these numbers by 7 and taking the integer portion yields 0 and 39, respectively. Thus the value 0 would be stored in location $9307. In addition, the value 39 would be stored in location $9308.

The top and bottom parameters are each divided by 8. In keeping with the Applesoft standard, the top row would be 0, and the bottom row would be 191. Dividing both of these values by 8 and taking the integer portion yields 0 and 23, respectively. In a similar manner, the value 0 would be stored in location $9309, and the value 23 stored in location $930A.

This technique of dividing the parameters by 7 or 8 significantly shortens the size of the screen dump subroutine. The one drawback, however, is that the screen area to be printed cannot be specified exactly. Instead, it is specified in blocks of 7 dots horizontally and 8 dots vertically.

Another feature of this subroutine is the ability to terminate the screen dump at any time simply by pressing the escape key. Pressing this key will return control to the calling program.

The following instructions show the steps required in creating a binary disk file containing this object code.

1. Protect memory locations above $92FF by setting high memory pointers to $92FF (37631). While in Applesoft type "**HIMEM: 37631**".
2. Enter the monitor by typing "**CALL -151**".

```
936B 4D 0D 93          EOR IMAGE
936E 20 9F 95          JSR PRCOUT
9371 E8                INX
9372 E0 07             CPX #$07        ; 7 BYTES/BLK
9374 D0 F2             BNE PCOL
9376 AD 0F 93          LDA COL
9379 CD 08 93          CMP RCOL        ; DONE WITH BLK ?
937C F0 06             BEQ NEXROW
937E EE 0F 93          INC COL
9381 4C 60 93          JMP PBLK
9384 20 98 95  NEXROW  JSR CRLF        ; OUTPUT CR, LF
               ;
               ; CALCULATE NEXT ROW
9387 AD 0E 93          LDA ROW
938A CD 0A 93          CMP BROW        ; DONE ?
938D F0 06             BEQ DONE
938F EE 0E 93          INC ROW
9392 4C 2D 93          JMP PROW
9395 4C 75 95  DONE    JMP RESETPR     ; RESET PRINTER
               ;
               ;************************************
               ;* DUMP THE HIRES SCREEN, EXPANDED *
               ;* VERSION.  EACH DOT ON SCREEN IS *
               ;* REPRESENTED BY 4 DOTS ON THE    *
               ;* PRINTER (2 DOTS X 2 DOTS).      *
               ;************************************
               ;
9398 A9 07     DUMPE   LDA #$07        ; LINE SPACING
939A 20 77 95          JSR LINESPC     ; = 7 DOTS
939D AD 08 93          LDA RCOL        ; START AT RIGHT
93A0 8D 0F 93          STA COL
               ;
               ; PRINT FAR RIGHT DOTS IN ONE COLUM
               ; OF BLOCK -- 7 COLUMNS OF DOTS
93A3 20 64 95  PCOLR   JSR CHKKEY      ; INTERRUPT ?
93A6 AD 0C 93          LDA LMARGE      ; LEFT MARGIN
93A9 20 89 95          JSR TAB
93AC 38                SEC             ; COMPUTER DOTS
93AD AD 0A 93          LDA BROW        ; IN ONE COL
93B0 ED 09 93          SBC TROW
93B3 8D 14 93          STA MULT1
93B6 EE 14 93          INC MULT1
93B9 A9 10             LDA #$10
93BB 8D 15 93          STA MULT2
93BE 20 44 95          JSR MULTPLY
93C1 20 B6 95          JSR PDOTS       ; TELL PRINTER DOTS IN COL
93C4 AD 16 93          LDA PROD
93C7 8D 18 93          STA DOTS        ; SAVE # DOTS
93CA 20 9F 95          JSR PRCOUT
93CD AD 17 93          LDA PROD+1
93D0 8D 19 93          STA DOTS+1
93D3 20 9F 95          JSR PRCOUT
93D6 AD 09 93          LDA TROW        ; START AT TOP
93D9 8D 0E 93          STA ROW
93DC 20 C0 94  PCOLR1  JSR CALCBLK     ; CALC ADDRESS
               ;
               ; CREATE EXPANDED BYTES FROM 4 MSB F
               ; BLOCK TABLE, THEN PRINT
               ;
93DF A2 00             LDX #$00
93E1 A9 00     PCOLR2  LDA #$00
93E3 8D 12 93          STA EXBYTE
93E6 3E 1A 93          ROL BLKTAB,X    ; DISCARD MSB
```

```
93E9 A0 03              LDY #$03
93EB BD 1A 93  PCOLR3   LDA BLKTAB,X
93EE 2A                 ROL
93EF 2E 12 93           ROL EXBYTE
93F2 3E 1A 93           ROL BLKTAB,X
93F5 2E 12 93           ROL EXBYTE
93F8 88                 DEY
93F9 C0 00              CPY #$00
93FB D0 EE              BNE PCOLR3
93FD 3E 1A 93           ROL BLKTAB,X
9400 2E 12 93           ROL EXBYTE
9403 20 9A 94           JSR PXBYTE     ; PRINT BYTE
9406 20 9A 94           JSR PXBYTE     ; TWICE !
9409 E8                 INX
940A E0 08              CPX #$08       ; DONW WITH BLK?
940C D0 D3              BNE PCOLR2
940E AD 0E 93           LDA ROW        ; DONE WITH COL?
9411 CD 0A 93           CMP BROW
9414 F0 06              BEQ PCOLR4
9416 EE 0E 93           INC ROW
9419 4C DC 93           JMP PCOLR1
941C 20 98 95  PCOLR4   JSR CRLF       ; PRINT CR, LF
               ;
               ; PRINT FAR LEFT DOTS IN ONE COLUMN
               ; OF BLOCKS -- 7 COLUMNS OF DOTS
               ;
941F 20 64 95  PCOLL    JSR CHKKEY     ; INTERRUPT ?
9422 AD 0C 93           LDA LMARGE     ; LEFT MARGIN
9425 20 89 95           JSR TAB
9428 20 B6 95           JSR PDOTS      ; TELL PRINTER DOTS IN COL
942B AD 18 93           LDA DOTS
942E 20 9F 95           JSR PRCOUT
9431 AD 19 93           LDA DOTS+1
9434 20 9F 95           JSR PRCOUT
9437 AD 09 93           LDA TROW       ; START AT TOP
943A 8D 0E 93           STA ROW
943D 20 C0 94  PCOLL1   JSR CALCBLK    ; CALC ADDRESS
               ;
               ; CREATE EXPANDED BYTES FROM 4 LSB F
               ; BLOCK TABLE, THEN PRINT
               ;
9440 A2 00              LDX #$00
9442 A9 00     PCOLL2   LDA #$00
9444 8D 12 93           STA EXBYTE
9447 3E 1A 93           ROL BLKTAB,X   ; DISCARD 4 MSB
944A 3E 1A 93           ROL BLKTAB,X
944D 3E 1A 93           ROL BLKTAB,X
9450 3E 1A 93           ROL BLKTAB,X
9453 3E 1A 93           ROL BLKTAB,X   ; GET 4 LSB
9456 2E 12 93           ROL EXBYTE
9459 A0 03              LDY #$03
945B BD 1A 93  PCOLL3   LDA BLKTAB,X
945E 2A                 ROL
945F 2E 12 93           ROL EXBYTE
9462 3E 1A 93           ROL BLKTAB,X
9465 2E 12 93           ROL EXBYTE
9468 88                 DEY
9469 C0 00              CPY #$00
946B D0 EE              BNE PCOLL3
946D 20 9A 94           JSR PXBYTE     ; PRINT BYTE
9470 20 9A 94           JSR PXBYTE     ; TWICE !
9473 E8                 INX
9474 E0 08              CPX #$08       ; DONE WITH BLOCK ?
```

3. Enter the binary code into locations $9300 through $95FF. For example, the first 8 bytes would be entered as follows:

9300:4C 22 93 4C AA 93 20 00

See page 44 of the *Apple II Reference Manual* for more detail.

4. Exit the monitor by typing a control-B followed by a return.

5. Save the object file to the disk by typing:

"BSAVE OBJ.DUMP,A$9300,L$2FF"

To use this subroutine as part of a BASIC program it is first necessary to protect all memory locations above $92FF (37631). In Applesoft this is done by the command **HIMEM: 37631.** Second, the object code must be loaded from the disk. In this case one would type **"BLOAD OBJ.DUMP,A$9300"**. The subroutine is now ready for use.

The BASIC Program

Listing 2 is the BASIC program which demonstrates the use of the screen dump subroutine. This program provides an easy means of loading, displaying and dumping HiRes graphics.

The program consists of five main parts. Lines 100 through 200 load the object code and initialize the screen dump parameters. Lines 210 through 350 prompt the user for the binary file name. Line 280 will then load the image into HiRes page 1.

The third section of this program, lines 360 through 710, allows the user to select different screen dump parameters. When first run, these parameters are set to the default values shown in Table 1.

Lines 730 through 800 pass the screen dump parameters to the machine language subroutine through the use of POKE statements.

The final section, lines 810 through 850, calls either the normal or expanded size screen dump subroutine. Once the screen dump is complete or terminated by pressing the escape key, control is once again returned to the Applesoft program.

Concluding Remarks

I have used this subroutine since August 1981, and have found it to work quickly and flawlessly. I am presently working on linking this machine language subroutine into an Apple Pascal library unit. This will improve upon the program published in Micro, February 1983.

```
9476 D0 CA              BNE PCOLL2
9478 AD 0E 93           LDA ROW         ; DONE WITH COL?
947B CD 0A 93           CMP BROW
947E F0 06              BEQ PCOLL4
9480 EE 0E 93           INC ROW
9483 4C 3D 94           JMP PCOLL1
9486 20 98 95  PCOLL4   JSR CRLF        ; PRINT CR, LF
               ;
               ; CALCULATE NEXT COLUMN
               ;
9489 AD 0F 93           LDA COL         ; DONE?
948C CD 07 93           CMP LCOL
948F F0 06              BEQ DONEE
9491 CE 0F 93           DEC COL
9494 4C A3 93           JMP PCOLR
9497 4C 75 95  DONEE    JMP RESETPR     ; RESET PRINTER
               ;
               ;************************************
               ;* PRINT BYTE                       *
               ;************************************
               ;
949A AD 12 93  PXBYTE   LDA EXBYTE
949D 4D 0D 93           EOR IMAGE
94A0 4C 9F 95           JMP PRCOUT
               ;
               ;************************************
               ;* ROTBLK - ROTATE BLK TABLE SO THA *
               ;* BITS ARE NOT ALIGNED FOR PRINT-  *
               ;* ING TOP OF IMAGE AT TOP OF PAPER *
               ;************************************
               ;
94A3 A0 07     ROTBLK   LDY #$07      ; 7 BITS
94A5 A2 07     ROTBLK1  LDX #$07      ; 8 BYTES
               ;
               ; MAKE BYTE FROM LSB BITS IN BLKTAB
               ;
94A7 5E 1A 93  MAKEBYT  LSR BLKTAB,X
94AA 6A                 ROR
94AB CA                 DEX
94AC E0 FF              CPX #$FF
94AE D0 F7              BNE MAKEBYT
94B0 48                 PHA             ; PUSH BYTE ON STACK
94B1 88                 DEY
94B2 D0 F1              BNE ROTBLK1
               ;
               ; MAKE NEW BLKTAB FROM 7 BYTES
               ; STORED ON STACK
               ;
94B4 A2 06     GETBLK   LDX #$06      ; 7 BYTES
94B6 68        GETBYTE  PLA           ; POP BYTE OFF STACK
94B7 9D 1A 93           STA BLKTAB,X
94BA CA                 DEX
94BB E0 FF              CPX #$FF
94BD D0 F7              BNE GETBYTE
94BF 60                 RTS
               ;
               ;************************************
               ;* CALCULATE BLKPT (BLOCK POINTER)- *
               ;* BLKPT IS THE ADDRESS OF THE TOP  *
               ;* BYTE IN THE BLOCK DEFINED BY ROW *
               ;* AND COL.                         *
               ;************************************
               ;
```

```
94C0 A9 00     CALCBLK  LDA #$00        ; BLKPT = SCRNPG
94C2 85 FC              STA BLKPT
94C4 AD 06 93           LDA SCRNPG
94C7 85 FD              STA BLKPT+1
               ;
               ; A= M0,M1,M3 OF ROW
               ;
94C9 A9 80              LDA #$80        ; BLKPT =
94CB 8D 14 93           STA MULT1       ; BLKPT+A:$80
94CE AD 0E 93           LDA ROW
               ;
               ; MASK OFF A
               ;
94D1 29 07              AND #$07        ; MASK = 00000111
94D3 8D 15 93           STA MULT2
94D6 20 44 95           JSR MULTPLY
94D9 18                 CLC
94DA A5 FC              LDA BLKPT
94DC 6D 16 93           ADC PROD
94DF 85 FC              STA BLKPT
94E1 A5 FD              LDA BLKPT+1
94E3 6D 17 93           ADC PROD+1
94E6 85 FD              STA BLKPT+1
               ;
               ; B= M4,M5 OF ROW
               ;
94E8 A9 28              LDA #$28        ; BLKPT=
94EA 8D 14 93           STA MULT1       ; BLKPT+B*$28
94ED AD 0E 93           LDA ROW
               ;
               ; MASK OFF B THEN SHIFT TO LSB POSIION
               ;
94F0 29 18              AND #$18        ; MASK = 00011000
94F2 4A                 LSR
94F3 4A                 LSR
94F4 4A                 LSR
94F5 8D 15 93           STA MULT2
94F8 20 44 95           JSR MULTPLY
94FB 18                 CLC
94FC A5 FC              LDA BLKPT
94FE 6D 16 93           ADC PROD
9501 85 FC              STA BLKPT
9503 A5 FD              LDA BLKPT+1
9505 6D 17 93           ADC PROD+1
9508 85 FD              STA BLKPT+1
950A 18                 CLC             ; BLKPT=
950B A5 FC              LDA BLKPT       ; BLKPT+COL
950D 6D 0F 93           ADC COL
9510 85 FC              STA BLKPT
9512 A5 FD              LDA BLKPT+1
9514 69 00              ADC #$00
9516 85 FD              STA BLKPT+1
               ;
               ;************************************
               ;* STORE 8 BYTES OF BLOCK (BLKPT) I *
               ;* BLOCK TABLE (BLKTAB). TOP BYTE   *
               ;* IS STORED IN BLKTAB,0            *
               ;************************************
               ;
9518 A5 FC     STOBLK   LDA BLKPT       ; INITIALIZE
951A 85 FE              STA TBLKPT      ; TBLKPT WITH
951C A5 FD              LDA BLKPT+1     ; BLKPT
951E 85 FF              STA TBLKPT+1
```

```
9520 A9 00                LDA #$00       ; START AT TOP
9522 8D 10 93             STA BLKROW     ; OF BLOCK
9525 A2 00       STOBYTE  LDX #$00       ; SAVE BYTE IN
9527 A1 FE                LDA (TBLKPT,X) ; TEMP LOC
9529 48                   PHA
952A AD 10 93             LDA BLKROW     ; SAVE TEMP
952D AA                   TAX            ; IN BLKTAB
952E 68                   PLA
952F 9D 1A 93             STA BLKTAB,X
9532 18                   CLC            ; TBLKPT=
9533 A5 FF                LDA TBLKPT+1   ; TBLKPT+$400
9535 69 04                ADC #$04
9537 85 FF                STA TBLKPT+1
9539 EE 10 93             INC BLKROW     ; COUNT BYTES
953C AD 10 93             LDA BLKROW     ; 8 BYTES
953F C9 08                CMP #$08       ; TRANSFERRED ?
9541 D0 E2                BNE STOBYTE
9543 60                   RTS
                 ;
                 ;************************************
                 ;* 8 * 8 BIT MULTIPLICATION         *
                 ;* PROD = MULT1 * MULT2             *
                 ;************************************
                 ;
9544 A2 08       MULTPLY  LDX #$08       ; MULT 8 BITS
9546 A9 00                LDA #$00       ; CLEAR MSB PROD
9548 8D 17 93             STA PROD+1
954B 0A          SHIFTL   ASL
954C 2E 17 93             ROL PROD+1
954F 0E 14 93             ASL MULT1
9552 90 09                BCC CHECKCT    ; CHECK BIT
9554 18                   CLC            ; ADD TO PROD
9555 6D 15 93             ADC MULT2
9558 90 03                BCC CHECKCT
955A EE 17 93             INC PROD+1
955D CA          CHECKCT  DEX
955E D0 EB                BNE SHIFTL     ; DONE ?
9560 8D 16 93             STA PROD
9563 60                   RTS
                 ;
                 ;************************************
                 ;* CHECK TO SEE IF ESC KEY HAS BEEN *
                 ;* PRESSED. RETURN TO CALLING PROG. *
                 ;* IF ESC HAS BEEN PRESSED, ELSE    *
                 ;* CONTINUE.                        *
                 ;************************************
                 ;
9564 AD 00 C0    CHKKEY   LDA KEYDATA    ; LOOK AT KEYBRD
9567 8D 10 C0             STA KEYSTRB    ; CLEAR STROBE
956A C9 80                CMP #$80       ; KEYPRESS ?
956C 90 06                BCC CONT
956E C9 9B                CMP #ESC       ; ESC KEY ?
9570 D0 02                BNE CONT
9572 68          STOP     PLA            ; PULL OFF
9573 68                   PLA            ; RETURN ADDRESS
9574 60          CONT     RTS
                 ;
                 ;************************************
                 ;* RESET PRINTER TO ORIGINAL LINE   *
                 ;* SPACING, 12 DOTS                 *
                 ;************************************
                 ;
9575 A9 0C       RESETPR  LDA #12
                 ;
                 ;************************************
                 ;* SET LINE SPACING TO NUMBER OF DOS *
                 ;* PASSED IN ACCUMULATOR            *
9577             ************************************
                 ;
9577 48          LINESPC  PHA
9578 20 98 95             JSR CRLF       ; PRINT CR, LF
957B A9 1B                LDA #$1B       ; ESC
957D 20 9F 95             JSR PRCOUT
9580 A9 41                LDA #$41       ; 'A'
9582 20 9F 95             JSR PRCOUT
9585 68                   PLA            ; # DOTS
9586 4C 9F 95             JMP PRCOUT
                 ;
                 ;**********************************
                 ;* SEND NUMBER OF SPACES, PASSED  *
                 ;* IN ACCUMULATOR TO PRINTER      *
                 ;**********************************
                 ;
9589 A8          TAB      TAY
958A C0 00       SPACE    CPY #$00
958C F0 09                BEQ RETTAB     ; RETURN
958E A9 20                LDA #$20       ; SPACE
9590 20 9F 95             JSR PRCOUT
9593 88                   DEY
9594 4C 8A 95             JMP SPACE
9597 60          RETTAB   RTS
                 ;
                 ;************************************
                 ;* OUTPUT A CARRIAGE RETURN AND A   *
                 ;* LINE FEED TO PRINTER             *
                 ;************************************
                 ;
9598 A9 0D       CRLF     LDA #$0D       ; ASCII CR
959A 20 9F 95             JSR PRCOUT
959D A9 0A                LDA #$0A       ; ASCII LF
                 ;
                 ;************************************
                 ;* SEND CHARACTER, PASSED IN ACCUM. *
                 ;* TO THE PRINTER. SLOT = 1         *
                 ;************************************
                 ;
959F 8D 90 C0    PRCOUT   STA PROUT      ; SEND CHAR
95A2 A9 00                LDA #$00       ; DELAY
95A4 8D 13 93             STA TIME
95A7 CE 13 93    DELAY    DEC TIME
95AA D0 FB                BNE DELAY
95AC AD C1 C1    WAIT     LDA PRWAIT     ; WAIT UNTIL
95AF 29 80                AND #$80       ; PRINTER READY
95B1 C9 00                CMP #$00       ; FOR NEXT CHAR
95B3 D0 F7                BNE WAIT
95B5 60                   RTS
                 ;
                 ;************************************
                 ;* SEND PRINTER ESC K TO SET UP TO  *
                 ;* ACCEPT NUMBER OF DOTS INFO.      *
                 ;************************************
                 ;
95B6 A9 1B       PDOTS    LDA #$1B       ; ASCII ESC
95B8 20 9F 95             JSR PRCOUT
95BB A9 4B                LDA #$4B       ; ASCII K
95BD 4C 9F 95             JMP PRCOUT
                 ;
95C0                      END
```

A Timely Interrupt

by Mike Hamilton

A most accurate timer for Standard Color BASIC, Extended Color BASIC, or ML programs can be made with just a few POKEs

The Color Computer probably has one of the most accurate and simple interrupt timers available on any microcomputer. All that is required to use it is understanding of a few points about the Color Computer hardware and software.

The Video Display Generator (VDG) displays an entire screen 60 times per second. After each screen display, the VDG toggles its horizontal synchronization line which is tied to the interrupt input of a Peripheral Interface Adapter (PIA). When bit zero of the control register of this PIA is set, the interrupt is enabled and is passed on to the Interrupt ReQuest (IRQ) pin of the 6809 MicroProcessor Unit (MPU). When the MPU receives this interrupt, it fetches the address of the IRQ routine from memory locations $FFF8 (65528) [high address byte] and $FFF9 (65529) [low address byte]. This ROM address is permanently set to point to RAM location $010C (268) which is a three byte JMP program that we can easily alter to fit our needs.

The idea behind using IRQ as a timer is simple: every interrupt from the VDG, add one to a 16-bit register, reset bit seven of the PIA control register to enable the next interrupt, and return to processing. Extended Color BASIC does something similar with its TIMER command. You can check by comparing the TIMER value with the 16-bit register at $0112 (274) and $0113 (275) with this program:

```
10 PRINT PEEK(274)*256+PEEK(275);
        TIMER:GOTO 10
```

Doing the same in Standard Color BASIC requires a bit more programming. Listing 1 is the short BASIC program. Line 10 reserves memory space for a short IRQ processing machine language routine that is POKEd into memory. The assembly listing for this interrupt processor is shown in Listing 2. Note that the Data Direction Register of the PIA at $FF03 (65283) must be read to reset the IRQ interrupt. This is done by the LDA $FF03. Line 20 changes the JMP instruction at $010C (268) to point to the new IRQ routine. Line 30 POKEs the ML routine into memory. Line 40 enables the IRQ interrupt. Line 50 is the DATA for the ML routine. Line 60 shows the timer operating.

Now, even if you erase the BASIC program, the timer will continue to operate. Resetting the computer will reset the interrupt enable bit to its normal value and the timer will stop. Another way of stopping either the 'homemade' timer or Extended BASIC's TIMER is to:

```
POKE(65283),PEEK(65283) AND 254
```

which will disable the IRQ interrupt. A timer is one of the simplest uses of the interrupt, but other tasks that require constant updating, such as printing a message in the corner of the screen, are also easily implemented. Proceed with caution though, because the more elaborate and time consuming the task, the slower the main program will function, especially in BASIC.

Listing 1 BASIC Timer

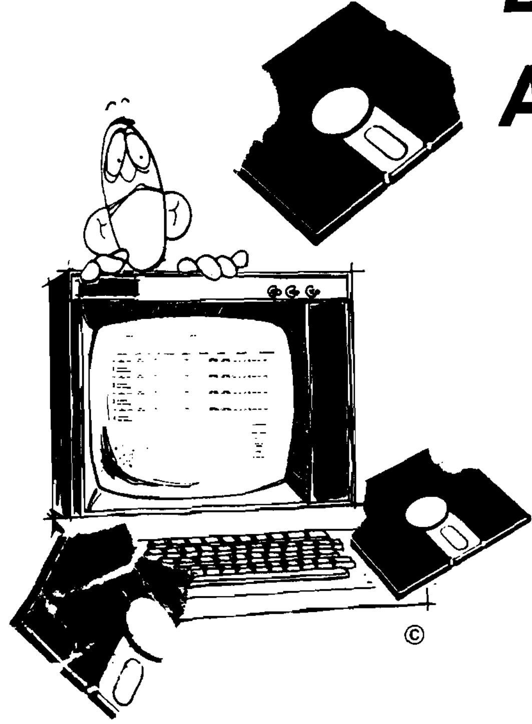
```
10 CLEAR 255,16367  REM  RESERVE ML SPACE
20 POKE 269,63:POKE 270,240 REM IRQ JMP VALUE
30 FOR I=1 TO 12:READ A:POKE 16367+I;NEXT I REM POKE ML
40 POKE 65283,PEEK(65283) OR 1  REM  ENABLE INTERRUPT
50 DATA 190,63,253,48,1,191,63,253,182,255,2,59
60 PRINT PEEK(16381)*256+PEEK(16382):GOTO 60
```

Listing 2 IRQ Handler

```
                    TIME    EQU   $3FFD   ; 16381

                            ORG   $3FF0

3FF0 BE 3FFD        START   LDX   TIME
3FF3 30 01                  LEAX  1,X     ; ADD 1 TO TIME
3FF5 BF 3FFD                STX   TIME
3FF8 B6 FF02                LDA   $FF02   ; READ DDR TO RESET
3FFB 3B                     RTI           ; RETURN
```

— Disk Abuse — A True Story

by Mark S. Morano



The sad confessions of a disk junkie. Beware lest you, too, fall upon this sad fate.

It is always uncomfortable broaching a subject such as this. People would rather not know, would like to believe it's just a story, a rumor, a bad dream. But we both know the truth - its a nightmare.

I should know - I've been there. I remember how it first started. You know that first time always sticks with you. It had been a long tense day at the office. We had this package to get out before a competitor, so we were working pretty hard. The pressure had been mounting and there were more bugs than mosquitos in a swamp. You know how it is - seems like they're breeding in an invisible subroutine somewhere. About 3 a.m. I was alone, huddled over my terminal when Error 13 - disk error, popped up on the screen. It always seems to creep out at the worst time. But now it had made its last interrupt. I was mad, raging like a bull. I slammed my fist on the desk, punched the disk drive open and ripped that diskette into a thousand pieces. There I sat, tracks and sectors everywhere. Then this strange sense of relief, a calming came over me. The kind of feeling you didn't want to let go of ... but, inexorably, it faded and was gone. I came to my senses and cleaned up the mess before I was discovered by the security guard, or worse yet - the night janitor. Still shaken, I powered down and called it a night.

At first I thought it was a one time thing; I wrote it off to nerves, a bad day, too much pressure. But then, it happened again. Another late night alone with my terminal. Feeling tired and tense, thinking about a drink, but knowing what I really wanted -- that release. And there was only one way I knew how to get it. Suddenly it was upon me. I found myself jumping up and down on a poor defenseless diskette.Trampling it to bits, I couldn't stop myself. After it was over, surrounded with cardboard and tape, that euphoric peace came over me. I was caught in an infinite loop.

No one knew for a while. A few suspected in Purchasing when the requests for diskettes started growing. A box or two soon deteriorated into cartons and cases. Being a group leader I put the blame on my subordinates, but I knew discovery was inevitable.

Then one day George came in to get a diskette I had borrowed. You can imagine the horror when he found it covered with teeth marks. I said it fell on the floor and I accidentally rolled over it with my chair. It was run over by a motorcycle, it ... he wasn't buying it. I confessed the truth was my dog got a hold of it. That was my slipup; George knew I didn't have a dog. He just shook his head and walked away.

My excuses wore thin - faulty drive, bad lot - people stopped believing. I found myself selling my peripherals to support what was now a $500 a week habit - diskettes don't come cheap these days.

Well, now I'm out a job, my wife and kids have left me and I sit around thinking up mail order scams -- you know, offering great games on diskette for five dollars if you send a diskette. They never see either again. And so it goes, on and on. Someday I'll be able to pass the local computer store without getting the shakes. But right now I can't; I just lie awake at night thinking of Winchester's. Its a hard way to go.

MICRO™

CoCo Bits

by John Steiner

As promised last month, we are going to take a closer look at **BASIC09**, and its advantages for the CoCo user. First, there is an omission in the documentation about loading BASIC09 that has caused a few problems for Tandy, and I am sure has frustrated some users. In the documentation, the only instruction to enable the user to access BASIC09 is to type 'basic09' ENTER. The user is then confronted with an ERROR 216 (file not found). Putting the BASIC09 disk in drive 1 and entering /d1/basic09 causes OS-9 to display ERROR 214 (file not accessable).

After some frustrating attempts at circumventing the problem, I finally resorted to calling my salesman at the Radio Shack store. As it turned out, I had called at just the right time, as his morning mail that day contained the solution to the problem in the form of a technical note from Tandy. In case you run into the problem with your OS-9, here is the simple solution. Either use the COPY command to copy BASIC09 into the command directory, so it will be accessable from command level, the way Tandy uses it in the BASIC09 manual, or use the LOAD command to load BASIC09 into OS-9's workspace:

```
COPY /D1/basic09 /D0/cmd/basic09
LOAD /D1/basic09
```

Either choice works, but using COPY to put BASIC09 into the cmd directory is the most convenient for future use. Create an OS-9 diskette with basic-09 in the cmd directory, and you will have it available as needed.

One of the questions I am most often asked is what will BASIC09 do that I can't do from Extended Color BASIC. Programming in the highly structured BASIC09 is quite a bit different from working with Color BASIC. BASIC09 is a much more powerful language and, if you learn it well, you will be able to create faster, more powerful operating programs. Currently, it is the closest thing to a full-fledged compiler CoCo users have access to, in that the packed modules are really compacted BASIC code executed from a run time package.

Programs written in BASIC09 are written in modules. Use the Edit mode to enter your program. Type E procedurename to open the edit file for your procedure. At this point, the first character of the line is expected to be an edit command. To insert a line in your procedure just type a space, the program line and the ENTER key. All program lines must begin with a space.

The other edit commands are + (go to next line), -(move back one line), L (list current line) and D (delete current line). One unique advantage of BASIC09 over Color BASIC is that error checking is done at the time each line is entered. Syntax errors, and similar line entry errors are trapped before you leave the line. In addition, an error check is done before leaving the procedure to check for undefined gotos, gosubs, etc. Also, if there is not enough memory for your procedure and any arrays, you will be warned at that point.

After typing a space, your program line can begin with a number if you want. If you plan to use GOTO or GOSUB to call the line you are typing, it will have to have a line number. Once you type a few lines you can list the procedure by typing L*. Your program will be listed with the hexadecimal I-code address next to each line. The I-code address is used to refer to individual program lines when an error occurs, or for other reasons.

BASIC09 has four modes. We have been working with the edit mode. The system mode is used to save, load, pack, rename and otherwise manipulate procedures from workspace to disk, or vice versa. The execution mode is entered whenever it is time to run your procedures. There is also an autorun feature that allows you to load and run programs from OS-9. If an error occurs, BASIC09 automatically enters the debug mode unless ON ERROR GOTO has been implemented.

Debug is one of the most powerful programming aids I have seen. It is even possible to execute the procedure one line of source code at a time. You can even display the source code on the screen while it is executing. Debug mode is very powerful, and has much more ability to assist with errors in programming than Extended Color BASIC.

BASIC09 will be a useful package for you, if you can justify the $170.00 or so that the language will add to the cost of your system. I hope to have some practical BASIC09 programs for you in the future. If you have hints or techniques regarding either BASIC09 or OS-9, send them in; we will pass them along.

New Tandy Drives for COCO

The TEC drive which Tandy has sold with the CoCo seems to have been replaced with a new unit Not many people I have talked with have been overly impressed with the performance level of the TEC drive, so this is good news. Tandy is now supplying the same drive unit that is found in the Model III and 4. A redesigned controller to work with the CoCo 2 has also been released, which doesn't require the 12 volt line that the CoCo 2 doesn't have at the cartridge port.

The new package should be a reliable addition to the CoCo line. The redesigned controller uses phase lock loop technology for data separation, which will also improve performance and stability. The only minus is Tandy's insistance on saving pennies by not gold plating the contacts on either the drive or controller cards. By the way, if you have a multi-pack interface, you can use the old drive controller with the CoCo 2. Also, Tandy still configures the cables, so if you are mixing drives be aware of the configuration situation.

Two Disk Utilities

I have received two utilities for review that are useful for the person who has to duplicate large quantities of software for production purposes. I am impressed with both of them.

Disk Manager by Elite Software contains two programs that allow copying to tape or disk Load either DTCOPY (tape) or DDCOPY (disk), and RUN. Insert the source disk in drive zero and press ENTER. The drive will read the directory and list the title of each entry by a number. Load the destination disk or tape, and enter the number of the program(s) desired. (e.g. 1,5,6, 12,14). The copy utility will copy only those files on the destination disk or tape. Entering ALL will cause the entire disk to be backed up. The disk version will make multiple copies with only one entry, and will offer Copy, and Abort options if a file is already on the destination disk. Other options include single drive operation, and rearranging the order of files on the destination diskette. The disk version requires installation of a formatted diskette.

FASTDUPE by Spectrum Projects allows duplication and formatting of an entire diskette. The only requirement is that the size of programs on the diskette must be small enough for all of them to fit in the 64K CoCo. FASTDUPE will first read the source disk and ask you to remove it. Install unformatted diskettes in drives 0 through 4 and press ENTER. FASTDUPE will then format and copy all four drives in succession, and let you install four more to do it again. If you don't have four drives, it will work with three, two, or even a single drive. Any bad copies are flagged, and the process continues. If you are just reproducing diskettes with a few small programs FASTDUPE will save you a lot of time.

MICRO

MICRO™ Interface Clinic

by Ralph Tenny

As promised in the last issue, we're going to look at programming the 6526 CIA (Complex Interface Adapter) I/O chips in the Commodore 64. This is a very complex IC which has a high capability and a correspondingly complex programming sequence to use all the CIA features. Here's a list of the I/O assignments for the two 6526s in the C-64:

U1 - Base Address $DC00

PA0 - PA7:	Keyboard Column Strobes Joystick B Paddle Multiplex
PB0 - PB7:	Keyboard Row Input Joystick A Fire Button/Light Pen
SR:	Shift Register # User Port
CNT:	Count Input User Port
PC:	Output Handshake Line Not used
FLAG:	Input Handshake/Interrupt Input Serial Bus
Timers (2):	System use
Time of Day Clock:	Available for User

U2 - Base Address $DD00

PA0 - PA1:	Memory Address Mapping
PA2 - PA3:	User Port
PA4 - PA7:	Serial Bus Control and Data
SR:	Shift Register #2 I/O User Port
CNT:	Count Input User Port
PC:	Output Handshake User Port
FLAG:	Input Handshake/Interrupt User Port
Timers (2):	Available for User
Time of Day Clock:	Available for User.

When you add it all up that is 16 User I/O lines. There are also two 9 Volt AC lines 5 VDC and four ground (power supply common) lines. Of these 16 I/O lines only PB0 - PB7 on U2 program in a completely straightforward manner. If you have the HESMON 64 machine language monitor cartridge or one of the several monitors available on disk (MINIMON, SUPERMON or others not from Commodore) you can follow this discussion more easily.

The B port is addressed at $DD01 and the B Data Direction Register (DDR) is at $DD03. The lines are set for output on a line-by-line basis. For example set Bit 0 of the DDR to logic 0 to make Bit 0 an input; otherwise set it to logic 1 for output. Once the direction assignments are made simply write 0 or 1 to output lines as needed or read input lines.

Turn on your C-64 and enter the monitor (with HESMON 64 plug in the cartridge and turn on power). Assign PB0-PB3 as input and PB4-PB7 as output by writing $F0 to $DD03. Now write 00 to $DD01 and try to read it back. What do you read? If there are no external connections made to the User Port you will read back $0F. The following lines illustrate that sequence as performed with HESMON 64. (User input appears in italics and the HESMON response in normal characters.) In HESMON memory modification is performed by positioning the cursor on a displayed memory value then entering the new value. In the display below this is shown by having the new entry immediately below the byte to be changed:

```
MDD00(ret)
:DD00 97 FF 3F FF FF FF FF
              F0(ret)
MDD00(ret)
:DD00 97 FF 3F F0 FF FF FF FF
          00(ret)
MDD00(ret)
:DD00 97 0F 3F F0 FF FF FF FF
```

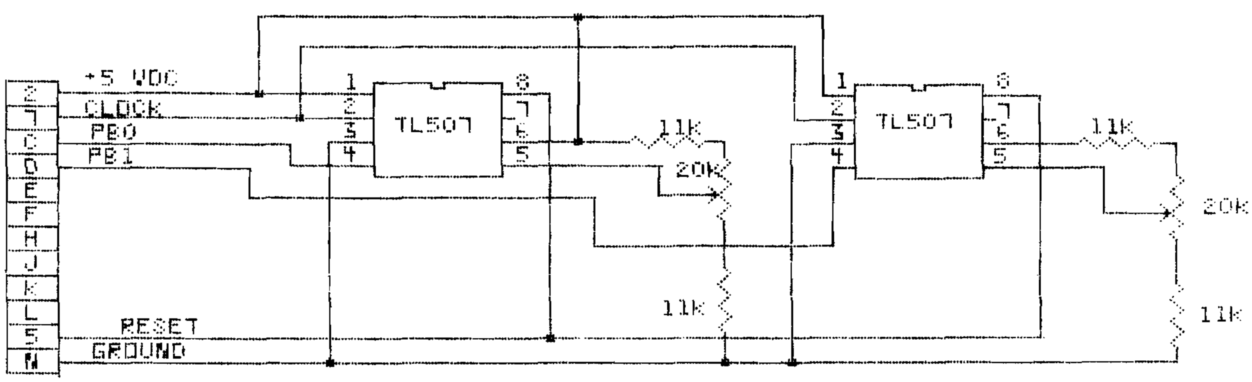
Although it is possible to accomplish the above experiment in BASIC the nature of PEEKs and POKEs will obscure the experiment's outcome. Programming the User Port lines PB0-PB7 in BASIC is possible but the rest of the lines are much more difficult to program in BASIC.

Listing 1 demonstrates the fundamentals of programming PA2 and PA3 of U2 and Listing 2 does the same for using the SP line in an assembly language program. Listing 1 assumes that Port A data direction assignments made during the C-64 power-up sequence have not been changed from $3F in $DD02. In fact careless modifications to location $DD02 can crash the computer as can **any** uninformed data manipulations involving U1.

The required sequence for controlling PA2 or PA3 of U2 is to set bits high with a logical OR and to set bits low with a logical AND operation. If you merely wish to change the logic level (toggle) the bit use an EXCLUSIVE OR with the same bit mask as the OR operation. Listing 1 lines 8 & 9 gives the OR bit mask which will set either PA2 or PA3 high. The proper instruction sequence to insure that PA2 is high is:

```
LDA #$04    ;BIT MASK FOR
ORA $DD00    ; BIT 2 = HIGH
STA $DD00
```

Figure 1



Listing 1

```
                 ; THIS PROGRAM DEMONSTRATES THE
                 ; ONLY SAFE WAY TO CHANGE PA2
                 ; AND PA3 OF CAI U2. USE THESE
                 ; LINE AS OUTPUTS ONLY. IT IS
                 ; ASSUMED THAT PORT A DATA
                 ; DIRECTIONS ASSIGNMENTS REMAIN
                 ; AS INITIALIZED BY BASIC.
                 ;
                 ; EQUATES
DD00              APORT    EQU $DD00   ; PORT LOCATIONS
DD01              BPORT    EQU $DD01
DD02              ADDR     EQU $DD02
```

```
DD03               BDDR    EQU $DD03
DD0E               TMRACR  EQU $DD0E
DD0F               TMRBCR  EQU $DD0F
                   ;
                   ; CONSTANTS
0004               A2TOG   EQU $04     ; BIT MASKS
0008               A3TOG   EQU $08
0040               SPTOG   EQU $40
                   ;
                   ; BUFFERS
007C               SAV1    EQU $7C
007E               REPCNT  EQU $7E
                   ;
C000                       ORG $C000
C000 A9 FF                 LDA #$FF    ; SET # OF REPEATS
C002 85 7E                 STA REPCNT
C004 A9 04         IN      LDA #A2TOG  ; INVERT PA2
C006 4D 00 DD              EOR APORT
C009 8D 00 DD              STA APORT
C00C A9 04                 LDA #A2TOG  ; INVERT PA2 AGAIN
C00E 4D 00 DD              EOR APORT
C011 8D 00 DD              STA APORT
C014 20 1C C0              JSR DELAY
C017 C6 7E                 DEC REPCNT  ; COUNT DOWN
C019 D0 E9                 BNE IN
C01B 00                    BRK         ; RETURN TO HESMON
                   ;
C01C 48            DELAY   PHA         ; SAVE REGISTERS
C01D 98                    TYA
C01E 48                    PHA
C01F 8A                    TXA
C020 48                    PHA
C021 A9 FF                 LDA #$FF    ; FULL COUNT
C023 8A                    TXA
C024 4A                    LSR A       ; HALF COUNT
C025 98                    TYA
C026 CA            SPIN    DEX         ; DELAY ROUTINE
C027 D0 FD                 BNE SPIN
C029 88                    DEY
C02A D0 FA                 BNE SPIN
C02C 68                    PLA         ; RETRIEVE REGISTERS
C02D AA                    TAX
C02E 68                    PLA
C02F A8                    TAY
C030 68                    PLA
C031 60                    RTS
C032                       END
```

Listing 2

```
                   ; THIS PROGRAM DEMONSTRATES
                   ; HOW TO TOGGLE THE SP LINES.
                   ; BIT 6 OF THE CONTROL REGISTER
                   ; PROGRAMS THE SHIFT REGISTER
                   ; DATA DIRECTIONS (0 = IN; 1 =
                   ; OUT). THE OUTPUT PIN GOES
                   ; TO LOGIC ONE WHEN SET FOR
                   ; INPUT (LOGIC ZERO FOR OUTPUT).
                   ;
                   ; EQUATES
DD00               APORT   EQU $DD00   ; PORT LOCATIONS
DD01               BPORT   EQU $DD01
DD02               ADDR    EQU $DD02
DD03               BDDR    EQU $DD03
DD0E               TMRACR  EQU $DD0E
DD0F               TMRBCR  EQU $DD0F
                   ;
                   ; CONSTANTS
0004               A2TOG   EQU $04     ; BIT MASKS
0008               A3TOG   EQU $08
0040               SPTOG   EQU $40
                   ;
                   ; BUFFERS
007C               SAV1    EQU $7C
007E               REPCNT  EQU $7E
                   ;
C000                       ORG $C000
C000 A9 FF                 LDA #$FF    ; SET # OF REPEATS
C002 85 7E                 STA REPCNT
C004 A9 40         IN      LDA #SPTOG  ; TOGGLE SP
C006 4D 0E DD              EOR TMRACR
C009 8D 0E DD              STA TMRACR
C00C A9 40                 LDA #SPTOG  ; TOGGLE SP AGAIN
C00E 4D 0E DD              EOR TMRACR
C011 8D 0E DD              STA TMRACR
C014 20 1C C0              JSR DELAY
C017 C6 7E                 DEC REPCNT  ; COUNT DOWN
C019 D0 E9                 BNE IN
C01B 00                    BRK         ; RETURN TO HESMON
                   ;
C01C 48            DELAY   PHA         ; SAVE REGISTERS
C01D 98                    TYA
C01E 48                    PHA
C01F 8A                    TXA
C020 48                    PHA
C021 A9 FF                 LDA #$FF    ; FULL COUNT
C023 8A                    TXA
C024 4A                    LSR A       ; HALF COUNT
C025 98                    TYA
C026 CA            SPIN    DEX         ; DELAY ROUTINE
C027 D0 FD                 BNE SPIN
C029 88                    DEY
C02A D0 FA                 BNE SPIN
C02C 68                    PLA         ; RETRIEVE REGISTERS
C02D AA                    TAX
C02E 68                    PLA
C02F A8                    TAY
C030 68                    PLA
C031 60                    RTS
C032                       END
```

Listing 3

```
                   ; THIS PROGRAM DEMONSTRATES HOW TO
                   ; USE TL-507 A/D CONVERTERS. IT
                   ; IS ASSUMED THAT EACH TL-507 IS
                   ; POWERING A POT AND READING THE
                   ; POT WIPER. THE POT POSITION IS
                   ; EXPRESSED AS A NUMBER BETWEEN
                   ; $00 AND $7F. PROVISION IS MADE
                   ; FOR CONTROLLING 8 TL-507 PARTS.
```

```
                 ; WITH THE RESULTING CONVERSIONS
                 ; DISPLAYED IN A LINE ACROSS THE
                 ; LOWER PART OF THE C-64 SCREEN.
                 ;
                 ; EQUATES
DD00              APORT    EQU $DD00    ; PORT LOCATIONS
DD01              BPORT    EQU $DD01
DD02              ADDR     EQU $DD02
DD03              BDDR     EQU $DD03
DC0E              SP1SET   EQU $DC0E
DD0E              SP2SET   EQU $DD0E
                 ;
                 ; CONSTANTS
0002              LIMIT    EQU 2        ; NUMBER OF A/DS TESTED
0040              SPINIT   EQU $40
0040              SPTOG    EQU $40
                 ;
                 ; BUFFERS
007C              SAVY     EQU $7C
007D              MASK     EQU $7D
007E              YINDEX   EQU $7E
007F              XINDEX   EQU $7F
0080              BUFFER   EQU $80      ; COUNT BUFFER AREA
0770              WINDOW   EQU $0770    ; FIRST DISPLAY LOCATION
DB70              WINCLR   EQU $DB70    ; COLOR RAM
                 ;
C000                       ORG $C000
                 ;
                 ; INIT PORTS AND PROGRAM
C000 A9 00        INIT     LDA #00      ; BPORT = INPUT
C002 8D 03 DD              STA BDDR
C005 8D 01 DD              STA BPORT
C008 A9 40                 LDA #SPINIT   ; SET SP LINES LOW
C00A 0D 0E DC              ORA SP1SET
C00D 8D 0E DC              STA SP1SET
C010 A9 40                 LDA #SPINIT
C012 0D 0E DD              ORA SP2SET
C015 8D 0E DD              STA SP2SET
                 ;
                 ; CLEAR COUNT BUFFERS
C018 A9 00                 LDA #00
C01A A2 07                 LDX #$07     ; INIT INDEX
C01C 95 80        WIPE1    STA BUFFER,X
C01E CA                    DEX
C01F 10 FB                 BPL WIPE1
                 ;
                 ; RESET A/DS
C021 A9 40        NEW      LDA #SPTOG   ; RESET LINE HIGH
C023 4D 0E DC              EOR SP1SET
C026 8D 0E DC              STA SP1SET
C029 A9 40                 LDA #SPTOG   ; THEN LOW AGAIN
C02B 4D 0E DC              EOR SP1SET
C02E 8D 0E DC              STA SP1SET
C031 AD 01 DD              LDA BPORT    ; GET INITIAL MASK
C034 85 7D                 STA MASK
C036 A0 00                 LDY #00      ; CLEAR COUNTER
C038 84 7C                 STY SAVY
                 ;
                 ; CLOCK A/D
C03A A9 40        CLOCK    LDA #SPTOG   ; CLOCK LINE HIGH
C03C 4D 0E DD              EOR SP2SET
C03F 8D 0E DD              STA SP2SET
C042 A9 40                 LDA #SPTOG   ; AND THEN LOW
C044 4D 0E DD              EOR SP2SET
```

This sequence modifies only PA2 leaving all other bits of Port A alone. To insure that PA2 is low use:

```
LDA  #$FB    ;ONLY BIT 2 IS LOW
AND  $DD00
STA  $DD00
```

Study the sequence of operations in Listing 1. This program toggles PA2 255 times with a delay between each operation. This allows you to monitor the action with a logic probe to verify the activity. Note in the DELAY subroutine that lines 26-30 save the A Y and X registers during the delay countdown and lines 39-43 restore the registers after the delay. This was not necessary **for this program's operation** but is good programming practice if you develop any routine which can be used as a mini-utility in all your programs.

Listing 2 is quite similar once you understand how to manipulate the SP line. This line is used to input or output 8 bits of synchronous serial data using the shift register internal to the 6526 CIA devices. If the Shift Register is set for input (Bit 6 of the CIA Control R Register A 0) the SP line goes high. Conversely programming the Shift Register for output (Bit 6 1) toggles SP low. Listing 2 toggles SP high then low 255 times with a delay between toggle operations. Note that exactly the same programming techniques can be used for SP1 (pin 5 of the User Port) by addressing $DC0E instead of $DD0E.

Both Listing 1 and Listing 2 were generated using the Commodore Assembler Development package (disk based) with intention of using HESMON 64 as a debugger. The BRK instruction (line 38) causes our program to stop by returning to HESMON. These examples will get you started on I/O programming on the C-64. We have not yet dealt with serial I/O using the Shift Register but we may get to that next time. Note also that PC and FLAG are not programmable directly. PC strobes low and back high automatically whenever Port B is written to or read from, furnishing an automatic handshake signal.

Whenever FLAG is pulled low bit 4 of the Control Register ($DC0D on U1 and $DD0D on U2) is set high. If the FLAG interrupt has been enabled an interrupt will be enabled. Otherwise you can poll this bit using:

```
LDA  $DD0D   ;GET INTERRUPT STATUS
AND  #$10    ;TEST BIT 4
BEQ  NOTHI   ;BRANCH TAKEN IF NO BIT
```

Your own code to process the bit received condition should follow directly.

Last month's experiment was a home-built single-slope A/D converter capable of operating from just two I/O lines. The TL507CP is a very low cost flexible A/D converter with 7 bits resolution (one part in 128) which is excellent to read pot or joystick position or two-wire sensors such as thermistors. Figure 1 shows a test circuit with two TL507s driven from the C-64 User Port. Up to eight TL507s can be controlled with this circuit and the results are displayed in a line across the lower one-fourth of the CRT. If fewer than eight A/Ds are connected all eight buffer locations will display but only those with data will change.

The TL507 is a single-slope A/D converter which contains a resistive ladder and a digital counter to generate the ramp. The ramp begins (count 0) at .75 Vcc and runs to maximum (count $7f) which occurs at 1.25 Vcc. Although this is inconvenient for converting DC voltages, potentiometers work very well. The TL507 works this way; the reset line (pin 8) is set high and then low. The output (pin 4) then switches high. Next the clock line (pin 2) is pulsed repeatedly until the output switches low. Just as in the experiment last time, the number of clock pulses required to switch the output is kept in a CPU register.

Listing 3 is the program which exercises the circuit of Figure 1. SP1 drives the Reset line, SP2 drives the Clock line, and the output lines of up to eight TL507s are sensed by PBO-PB7. It works this way: the TL507s are reset by lines 58-60 and the output lines all go high. This condition is stored in MASK and a counter is initialized. The clock lines are pulsed (simultaneously) one time and the post input pattern is compared to MASK (lines 67-76). If any TL507 output changes, lines 77-82 detect the change and save this new pattern in MASK. Lines 83-91 identify the TL508 which signalled Conversion Complete, save the clock count and display the count. This process is repeated until all input lines have been switched low, or until 127 clock pulses have been issued. Lines 93-104 control the display process. If fewer than 8 TL507s are connected, data buffers associated with the missing converters are reported as ''00''

```
C047 8D 0E DD           STA SP2SET
C04A A5 7C              LDY SAVY    ; GET COUNT
C04C C8                 INY         ; COUNT CLOCK PULSES
C04D 84 7C              STY SAVY    ; AND REMEMBER COUNT
C04F 30 D0              BMI NEW     ; START OVER IF COUNT >127
C051 AD 01 DD           LDA BPORT   ; READ A/D OUTPUTS
C054 45 7D              EOR MASK    ; TEST FOR A/D DONE
C056 F0 E2              BEQ CLOCK   ; NONE? CLOCK AGAIN
C058 AE 01 DD           LDX BPORT   ; READ PORT AGAIN
C05B 86 7D              STX MASK    ; MAKE THIS NEW MASK
C05D A2 00              LDX #00     ; CLEAR INDEX
C05F 4A        BITID    LSR A       ; TEST WHICH BIT HIGH
C060 90 08              BCC NXTBIT  ; IF TRUE, NOT THIS BIT
C062 94 80              STY BUFFER,X  ; IT WAS TRUE, SAVE COUNT
C064 20 71 C0           JSR SHOW    ; DISPLAY COUNTS
C067 4C 3A C0           JMP CLOCK   ; TEST OTHER A/DS
C06A E8        NXTBIT   INX         ; COUNT SHIFTS
C06B E0 02              CPX #LIMIT  ; ALL BITS TESTED?
C06D F0 B2              BEQ NEW     ; ALL DONE, START OVER
C06F D0 EE              BNE BITID   ; ELSE TEST NEXT BIT
               ;
C071 86 7F     SHOW     STX XINDEX  ; SAVE VALUES
C073 84 7E              STY YINDEX
C075 A2 00              LDX #00     ; CLEAR INDEX REGS
C077 A0 00              LDY #00
C079 B5 80     READ     LDA BUFFER,X  ; GET COUNT VALUE
C07B 20 88 C0           JSR OUTPUT  ; DISPLAY IT
C07E E8                 INX         ; POINT TO NEXT
C07F E0 08              CPX #08     ; TEST FOR LAST
C081 90 F6              BCC READ
C083 A6 7F              LDX XINDEX  ; RETURN WITH DATA
C085 A4 7E              LDY YINDEX
C087 60                 RTS
               ;
C088 48        OUTPUT   PHA         ; SAVE DATA
C089 4A                 LSR A       ; GET HIGH NIBBLE
C08A 4A                 LSR A
C08B 4A                 LSR A
C08C 4A                 LSR A
C08D 20 9E C0           JSR CONVRT  ; MAKE DISPLAYABLE CHAR
C090 20 AB C0           JSR DISPLY  ; SHOW IT
C093 68                 PLA         ; GET DATA AGIAN
C094 29 0F              AND #$0F    ; MASK TO LOW NIBBLE
C096 20 9E C0           JSR CONVRT
C099 20 AB C0           JSR DISPLY
C09C C8                 INY         ; SPACE BETWEN BYTES
C09D 60                 RTS
               ;
C09E C9 0A     CONVRT   CMP #$0A    ; ALPHA OR DIGIT?
C0A0 90 04              BCC NUMBER  ; 0 - 9
C0A2 38                 SEC         ; A - F
C0A3 E9 09              SBC #$09    ; MAKE IT C-64 SCREEN CODE
C0A5 60        EXIT     RTS
               ;
C0A6 18        NUMBER   CLC         ; CONVERT TO ASCII
C0A7 69 30              ADC #$30
C0A9 D0 FA              BNE EXIT    ; BRANCH ALWAYS
               ;
C0AB 99 70 07  DISPLY   STA WINDOW,Y  ; PUT IN SCREENBUFFER
C0AE A9 00              LDA #00     ; CHAR. COLOR = BLACK
C0B0 99 70 DB           STA WINCLR,Y  ; UPDATE COLOR RAM
C0B3 C8                 INY         ; BUMP INDEX
C0B4 60                 RTS
C0B5                    END
```

Commodore Compass

by Loren Wright

New Commodore Computers?

Commodore's CES announcement of two new computers was at least partially withdrawn. It appears now that the 264, if it appears at all, will be introduced late in the year. The 364 has been indefinitely postponed. It's probably just as well. I, and a number of others, doubted the wisdom of bringing out a whole new line just when the Commodore 64 had become established. The Commodore 64 finally has a respectable assortment of software available, and it is doing very well, I might add. The 1701 processor would have been the biggest hurdle. It would have taken a while to convert a significant amount of 6502/6510 software, and Commodore would have started again with the same problem it has always had with new computers - little, if any software!

Side Scrolling Update

Because of space limitations in last month's issue, you may have been left a little in the dark regarding how to use the side scrolling routine. What the routine does is move the screen contents, along with the corresponding color memory, to the right or left. If the move is to the left, then column 1 (actually the 2nd column) is copied into column 0. Column 2 is copied into column 1, and so on, until the move is complete. If the move is to the right, then column 38 is copied into column 39, column 37 into 38, and so on, until the move is complete. The program allows you to specify a range of columns to be moved. The left column (LCOL) must be POKEd into 49152, and the right column (RCOL) must be POKEd into 49153. On a left move LCOL must not be less than one, and on a right move RCOL must not be greater than 38. If there is a 0 in location 49154, then the last column copied will remain unchanged--i.e., there will be two identical columns adjacent. Most of the time, you will want that last column replaced with spaces, and any number besides 0 POKEd into 49154 will accomplish that result. The left move is called with SYS 49155, and the right move is called with SYS 49182.

It is a simple matter to add this feature to the screen editor (MICRO 66:28). In addition to the subroutine provided last month (70:59), only three lines are required:

```
5 GOSUB 19000: LC49152: RCLC1: POKE LC2,1
361 IF T$[ THEN GOSUB 1000: POKELC,1: POKE RC,H: SYS49155: GOTO200
362 IF T$] THEN GOSUB 1000: POKELC,H: POKE RC,38: SYS49182: GOTO200
```

The horizontal cursor position (H) is used to determine the end of the screen move. This is the quick-and-dirty implementation. It removes the two square brackets characters from use in a graphic, though. To get them back, I would suggest using one of the unused function keys (f4 and f8) to enter a command mode, which expects another key to complete the command. This allows for future expansion, such as up and down screen moves, fill routines, etc. To avoid errors, it would be a good idea to have some audible or visible (flashing border?) indication that another key is expected.

Communications Update

I was serious about including bulletin board listings and information in this column. So far I only have one such item (coming up next). I will also be checking my CompuServe EMAIL regularly, so for those who missed it, my CompuServe number is 70626,636. I won't always be able to give direct responses to questions, but information and news that you think would be of general interest is welcome.

TPUG BBS New Number and Policies

The Toronto PET Users Group (TPUG) has a new number for its bulletin board service: (416)-429-6044, 24 hrs, 7 days. The biggest change in policy is that users will no longer be able to download programs from the club library. To get library programs you must purchase the club's library disks or cassettes. However, the board may be used to upload programs, and this use is encouraged. The club librarians will go through programs so received regularly. Acceptable ones will be added to the library, and the contributor of an accepted program will get to choose a free library disk.

TPUG Conference

The Third Annual TPUG Conference will be held May 26 and 27, 1984, at the Constellation Hotel in Toronto. Features of the conference include two full days of lectures, workshops, and panel discussions conducted by local, as well as out-of-town, experts. The preliminary schedule shows at least five different speakers going at once all day Saturday and Sunday. Typical topics: Evaluating Commercial Software, Speech Synthesis, Hi-res Graphics on the C-64, Networking, How to Use Spreadsheets, and a Computer Music Overview.

Some of the more popular sessions, such as Jim Butterfield's day-long machine-language workshop, may be filled, but there should still be a lot to choose from. I enjoyed participating last year, and look forward to it again this year. My topics will be Sprite Programming Techniques (intermediate level), and C-64 Graphics: A Little Machine Language Goes a Long Way (intermediate/advanced).

Other activities at the conference include easy availability of copies of club library disks, an exhibit area for hardware and software vendors, an answer room, a trader's corner, and an optional banquet.

If you've never been to Toronto, I should tell you that it is a beautiful city with a lot going on! Registration (required to participate in events) is $25. In addition, you must be a club member, which costs $30 (regular) and $20 (associate). Associate membership is intended for out-of-town members. You still receive the club's magazine TORPET and have access to the club library. The number to call for more information is (416)-782-9252 (business hours only).

From Here to Atari

by Paul S. Swanson

I recently added an Atari 800xl computer to my collection. The main differences between that system and the earlier Atari 400, 800 and 1200xl computers is that no BASIC cartridge is required. BASIC is built in. I also noted some differences in the keyboard. The Atari 1200xl keyboard is still the best of the series and I use that system for my word processing. However, the Atari 800xl keyboard is close competition. The keys are non-glare type finish and have shorter strokes than the ones on the Atari 1200xl computer, which may be preferable to some touch typists.

What really impressed me about the Atari 800xl computer was one of the details of the design. The cartridge slot has two metal strips forming a double door configuration, opening inward when a cartridge is inserted. When the cartridge is removed they spring closed again. The impressive detail is that there is no way to trap your finger in it. This seems like a minor point until you consider having a small child at the computer. The configuration of the cartridge door looks like an ideal setup for trapping small fingers, but after spending about 20 minutes studying the door, I concluded that there was no way it could trap anything.

MYDOS UPDATE

Last month I reported a few bugs in MYDOS. Since then I have been in communication with SWP concerning that product. The updating of random access files has been corrected in versions 3.012 and 3.17. I noted that other minor bugs that I had uncovered have also been corrected. The new version should be available by the time you read this column.

I also acquired an 80-track disk drive for my ATR8000. Using MYDOS to configure it, the 80-track double sided Qume disk holds about 734K of usable disk space. That is a little more than eight times the capacity of an Atari 810 disk drive.

TELECOM UPDATE

Nite Lite, the computer bulletin board I'm running every night, has been a good source of information concerning how people are setting their Ataris up for telecommunication. If you have an Atari 850 interface or an ATR8000 you still have the widest selection. Any RS-232 compatible modem will connect to either of these devices directly.

There are many callers who do not have Atari 850 interfaces or ATR8000's connected to their systems. These Atari owners use either the Atari 835 modem or the MPP-1000. The Atari 835 is a little more expensive, but connects along the serial bus like other peripherals. The MPP-1000 plugs into a joystick port.

I have noted one problem with the MPP-1000C, which is that it doesn't respond as device R: so no custom software or any other software not specifically written for that modem will work. That eliminates what seems to be the most popular software on the Atari computer in this area, which is a public domain program called AMODEM. However, the MPP-1000 comes with software that is at least comparable. The problem arises when you want to do other things with the modem. For example, there are several people who want to start their own computer bulletin boards, which requires different software.

If you are looking for ways to get into telecommunication with your Atari computer, without an Atari 850 interface module or an ATR8000, the MPP-1000 is the least expensive route. Other than that one problem, I have heard no complaints, so that modem seems to be worthy of consideration. I will be looking into the features of that modem and ways around the problem of interfacing it to other software. This will be reported in future columns.

Information such as this can also be found on Nite Lite if you already have telecommunications capabilities at either 300 or 1200 baud. Nite Lite operates from 7:00 pm until 7:00 am, eastern time, at (617) 576-2426. If you call, leave me a message telling me that you got the number from this column. You are, of course, welcome to leave suggestions of issues for me to address in this column on Nite Lite.

Telecommunications is a rapidly expanding area on personal computers. As the number of callers increases, the amount of information and entertainment available from these computer bulletin boards increases proportionally. There are also other new services opening up that are accessible using the same equipment and software required to access the free bulletin boards.

MICRO

A Perfect 2nd Computer for the Apple Owner

HAVAC (Home/Academic Very Affordable Computer) is a transportable (14.2 lbs), 64K RAM, 40 column computer system compatible with the Apple II family. Its designed around the 6502 chip and a new 5.25", 164K disk drive. A stand alone drive is also offered as an expansion product.

Over 1000 of the most popular Apple programs have been successfully run on the system. Each HAVAC is shipped with an updated list of tested programs, and any special instructions needed to run them. These programs include games, education and business software.

This computer is aimed at first time users, but its low price of $850 also makes it the perfect 2nd computer for the two-computer family. That price includes 64K RAM, 8K ROM; 164K floppy disk drive; 62 Key detached keyboard supporting upper and lower case and 4 cursor keys, HiRES color graphics; printer port, serial port, game port and video hookup. Free software includes HAVAC DOS, Typewriter, Card File, Calculator, Utilities, HAVAC BASIC and HAVACOM.

MicroSci
2158 S. Hathaway Street
Santa Ana, CA 92705
714/241-5600

Three Useful Commodore Devices

The C-64 is missing an important feature...*a reset switch*. The only way to regain control on a hung-up computer is to turn it off and lose the data entered already. A reset switch is available which attaches with two simple solder connections, either externally in a separate box or through a hole drilled in the computer cover. With an enclosed software program, it allows recovery of entered data, and costs only $9.95.

An Interference Filter Kit solves the problem of RF emissions from the computer unit of older 64's. It requires opening the computer and making three simple solder connections, but can be done in minutes. Price is $19.95.

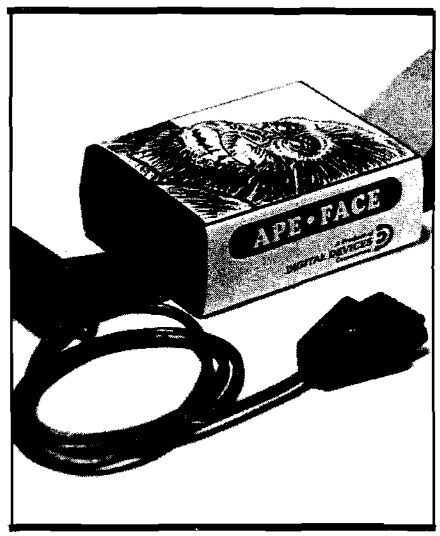
A Monitor Cable with 5 – Pin Din Plug with attach Commodore, Atari and other computers to the new 1702 Commodore Monitor. The cable packed with the monitor has an 8-pin din plug that won't work with many machines. This retails for $24.95.

Bytes & Pieces
550 N. 68th Street
Wauwatosa, WI 53213

Inexpensive Atari Printer Interface

APE-FACE allows Atari computer owners to choose from a variety of standard printers, including Epson, Okidata, Centronics and many other parallel style printers. There are currently two models: the 48P works with the Atari 400 and 800 computers; the XLP is for all Atari computers including the new XL line. There is no installation needed; simply plug in the cable to the printer and computer. The suggested price is $89.95.

Digital Devices Corp.
151 Sixth Street
Suite 127, O'Keefe Bldg.
Atlanta, GA 30313
404/872-4430

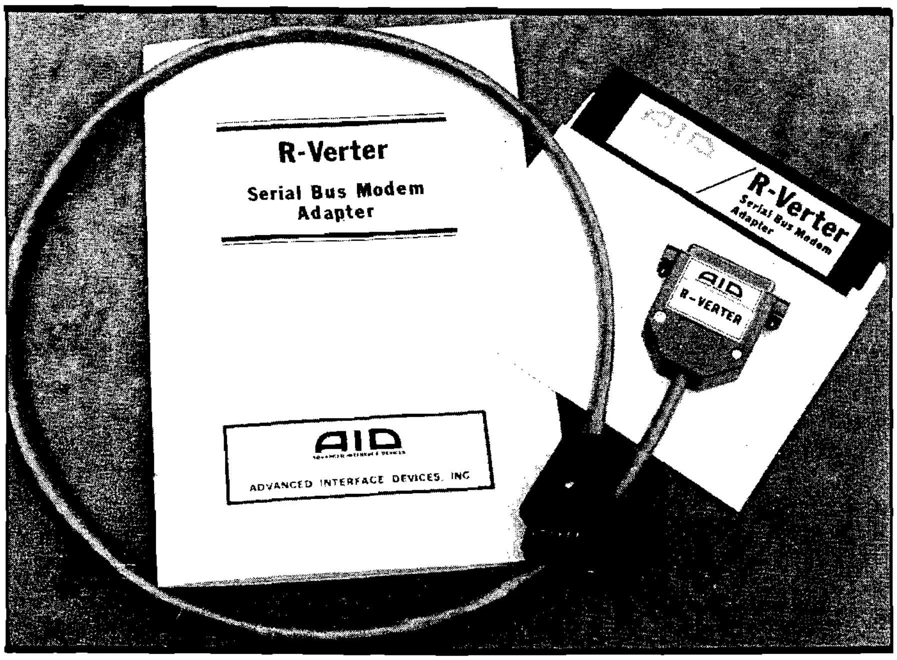


Modem Adapter for the Atari Serial Bus

The R-Verter, Serial Bus Modem Adapter for Atari 400, 600XL, 800 and 800XL home computer systems allows most modems and other RS-232C devices to be used directly without using the Atari 850 Interface Module or other interfaces. It comes with a software package which includes a smart terminal emulator and an RS-232C device handler, and will work with any RE-232C device which will accept TTL-level inputs (the majority will). The R-Verter requires no modifications of the computer or other peripherals and it does not use up a joystick port. All circuitry is contained in an RS-232C type connector to minimize size.

It comes with a built-in 3 foot cable and is available in either male or female connector configurations. When used with the A.I.D. Interfast-1 buffered printer interface (not included), it allows modem or RS-232C data to be echoed to a printer without first storing to a cassette or disk. Most common RS-232C handshaking configurations are available using internal jumpers. Price with terminal and print echo software is $49.95.

Advanced Interface Devices, Inc.
P.O. Box 2188
Melbourne, FL 32902
305/676-1275

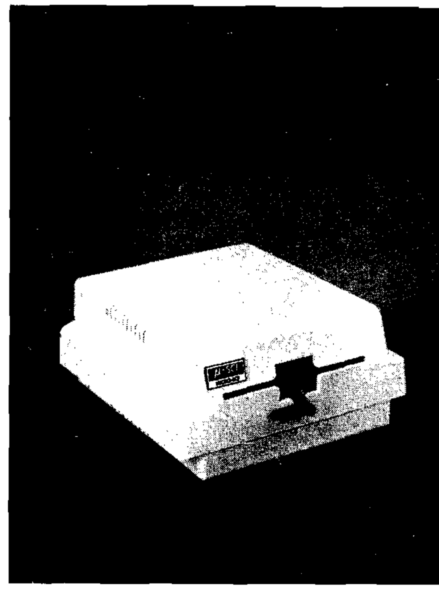


Apple Compatible Disk Drives Designed for End User

The XL and XL80 disk subsystems are Apple compatible floppy disk drives with capacities of 164K-bytes and 328K-bytes respectively. With access times of 18msec, the 40 and 80 track drives are useful for applications ranging from software requiring 35 track, 100% Apple compatibility on the small drive, to CP/M, PRODOS and volume sensitive software on the larger drive.

The drives are different in size and shape from typical OEM drives. This is largely due to the horizontal clutch carrier plate which is lowered via a nylon coated, miniature steel cable and activated by turning an ergonomically designed knob. The new drives are packaged in plastic housings and are slightly larger than standard OEM drives. According to the manufacturer, product maintenance and cost of maintenance is low due to fewer parts and simpler manufacturing process. End user pricing for the XL and XL80 are: $199 and $299.

MicroSci
2158 S. Hathaway Street
Santa Ana, CA 92705
714/241-5600



Parallel Interface Card for Apple

The UniPrint card is an easy-to-use and inexpensive interface card for parallel printers. It is compatible with the Apple II+, Apple IIe, and a wide variety of printers. The purchase price of $89.00 includes a Centronics compatible cable and graphic transfer capabilities.

UniPrint provides transfers of HiRes graphics pages one and two, expands and shrinks the images, or rotates the images in any direction by 90 degrees. Color transfers are also possible on the Dataproducts (IDS) Prism printer.

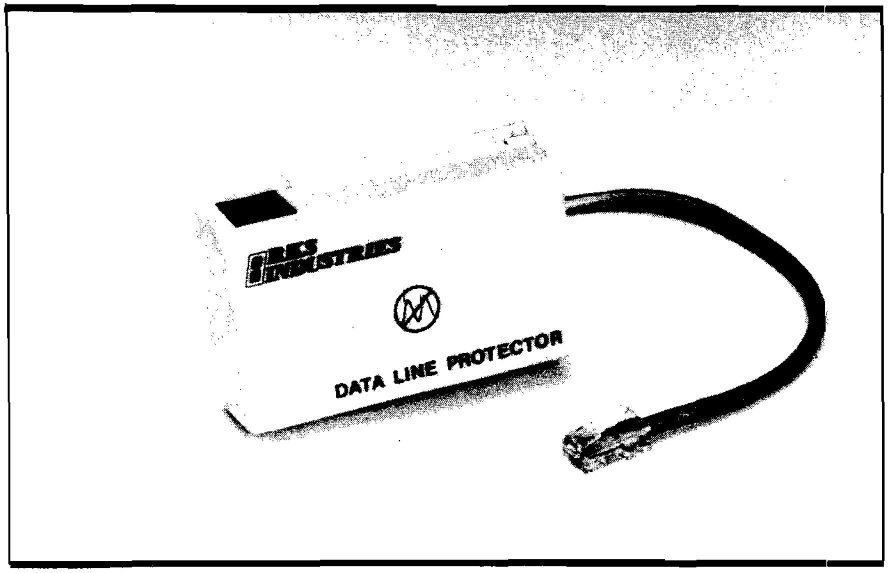
The manual provides a step-by-step procedure for installation of the UniPrint and even includes pre-tested configurations for the most popular parallel printers avialable. Over 25 printers are listed, including: Epson, C-ITOH, Apple DMP, Anadex.

Videx, Inc.
1105 N.E. Circle Blvd.
Corvallis, OR 97330
503/758-0521

Data Line Surge Protection

The SurgeSentry offers data line protection against power-induced problems such as static, electrical storms and other electrical interference problems. When a modem transmits its data signal via long distance phone lines, high voltage spikes and transients caused by storms, power stations, etc. can't be avoided and are carried along with the data being transmitted.

The data line protector constantly monitors the line as a passive device and, when a voltage spike is detected, clamps on the surge and absorbs the power from the line. This two-stage



suppression device was designed to provide the fastest reaction time and the highest absorption level, using silicon avalanch diodes and gas discharge tubes (spark gaps).

The SurgeSentry plugs directly into a household phone jack, and the modem cable plugs into it, thus protecting the line to the modem and equipment connected to it. The SurgeSentry retails for $89.50.

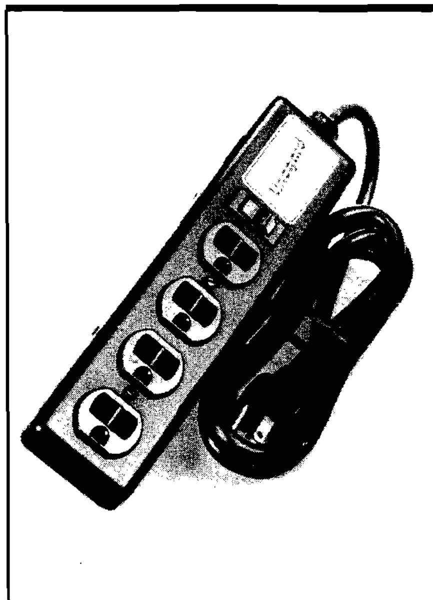
RKS Industries
4865 Scotts Valley Drive
Scotts Valley, CA 95066
408/438-5760

Surge Suppressor Outlet Strip

The LG20 Surge Suppressor Multi-Outlet Strip offers small computer owners protection against voltage surges that can damage and even destroy electronic solid state components. Since this product can easily be installed by simply plugging into any 15A125V AC outlet, it is ideal for home, business or office use.

This UL-listed 9 3/4" product features four "U ground" outlets, an on/off switch with pilot light, a six foot cord with three prong grounding plug, and a push-to-reset circuit breaker which protects against power overloads. The suppression circuit acts as a shock absorber by limiting surges or spikes without interfering with normal current flow. The LG20 will absorb surges up to 6000 volts or 6500 Amps in less than 10 nanoseconds. The unit limits voltage to a safe 205 volts. The LG20 sells for $34.95 plus $2 shipping and handling.

Gadgeteer
1524 Pine Street
Philadelphia, PA 19102
215/732-0965



NOTE: When you contact the manufacturers about these products, please be sure to tell them "I saw it in MICRO." We appreciate your support.

MICRO

Index of Reviews

June 1983 through May 1984

Title	Manufacturer	Type	Micro	Issue
Magic Memory	ARTSCI	A	APPLE	71
Apple Record Manager	CONN. INFO SYSTEMS	A	APPLE	68
Personal Finance Manager	APPLE COMPUTER INC	A	APPLE	67
File-Fax	TMQ SOFTWARE	A	APPLE	64
TGS: The Graphic Solution	ACCENT SOFTWARE	A	APPLE	63
Master Grades	MIDWEST SOFTWARE	A	AP/PET	64
Form Letter Writer	MMG MICRO SOFTWARE	A	ATARI	70
Mail List	MMG MICRO SOFTWARE	A	ATARI	68
Data Manager	MMG MICRO SOFTWARE	A	ATARI	67
Career Counselor	MMG MICRO SOFTWARE	A	ATARI	66
Homebase	HOMEBASE COMP. SYS.	A	COCO	70
C.C.Calc Disk Version	TRANSFORMATION TECH	A	COCO	67
Disk Data Handler	CUSTOM SOFTWARE ENGIN.	A	COCO	61
Semi-Draw	COMPUTERWARE	A	COCO	61
Mail controller	ORBYTE	A	C64	71
Diary 64	COMPUTER MARKETING	A	C64	68
Passive Solar Design for Home	DON DANVLYK	A	C64	67
64 Mail List	COMPUTER MARKETING	A	C64	66
Hello Central!	HOWARD W. SAMS	C	APPLE	63
Disk COLORCOM/E Smart Term.	EIGEN SYSTEMS	C	COCO	67
The Color Connection	COMPUTERWARE	C	COCO	65
BugByter	COMPUTER-ADVENTURE	D	APPLE	68
Ultra Disassembler	ADVENTURE INTER.	D	ATARI	66
BASIC Debugger	MMG MICRO SOFTWARE	D	ATARI	66
Atari BASIC Compiler	DATASOFT	D	ATARI	65
ABC	MONARCH DATA SYS	D	ATARI	63
DISKEDT	SPECTRAL ASSOCIATES	D	COCO	70
STARDOS 64	STAR KITS	D	COCO	70
BASIC Aid	EIGEN SYSTEMS	D	COCO	68
CCREAD Editor/Assmb/Debugger	EIGEN SYSTEMS	D	COCO	68
Disassembler for 6809	GRANITE COMPUTER SYS	D	COCO	67
Platinum Worksaver	PLATINUM SOFTWARE	D	COCO	62
Ultra 80CC	SPECTRAL ASSOCIATES	D	COCO	61
Computer Mechanic	SOFTSYNC	D	C64	71
SYSRES	SOLIDUS INTERNATIONAL	D	C64	67
Delta Drawing	SPINNAKER SOFTWARE	E	APPLE	71
Exper. in Human Physiology	HRM SOFTWARE	E	APPLE	71
Flight Simulator II	SUBLOGIC	E	APPLE	71
Word Attack	DAVIDSON & ASSOCIATES	E	APPLE	65
Watchwords & Wordisk Maker	MICROMEDIA SOFTWARE	E	APPLE	64
Bumble Games	THE LEARNING CO.	E	APPLE	64
General Chemistry	COMPRESS	E	APPLE	64
Face Maker	SPINNAKER SOFTWARE	E	APPLE	63
Multiploy	RESTON PUBLISHING CO	E	APPLE	62

Type Key: Application, Communication, Development, Educational, Forth, Game, Hardware, Language, Software, Tutorial, Utility or Word processor.

Title	Manufacturer	Type	Micro	Issue
Earl's Word Power: Homonyms	GEORGE EARL	E	APPLE	61
Decimal Practice	CONTROL DATA PUBL.	E	ATARI	70
Preparing for the SAT	PROGRAM DESIGN, INC	E	ATARI	67
Square Pairs	SCHOLASTIC INC	E	ATARI	67
Turtle Tracks	SCHOLASTIC INC	E	ATARI	67
Wordrace	DON'T ASK SOFTWARE	E	ATARI	67
Mathmenu 1.0	INTER+ACTION	E	COCO	70
Fundamentals of Mathematics	STERLING SWIFT PUBL.	E	C64	67
C64-FORTH	COMPUTER MARKETING	F	C64	67
Mickey in the Great Outdoors	WALT DISNEY	G	ATARI	66
Shrink	STAR KITS	G	COCO	68
Printmate 99 Printer	MICRO PERIPHALS	H	ALL	70
Printmate 150G Printer	MICRO PHERIPHALS	H	ALL	68
RAM/EPROM Memory Board	JOHN BELL ENGINEERING	H	ALL	62
APPLE Pin Saver	KEN BRANSCOME ASSOC.	H	APPLE	70
Ultra ROM Board/Editor	HOLLYWOOD HARDWARE	H	APPLE	67
KoalaPad Touch Tablet	KOALA TECHNOLOGIES	H	APPLE	67
KoalaPad	KOALA TECHNOLOGIES	H	ATARI	70
HJL-57 CoCo Replacement Kbrd.	HJL PRODUCTS	H	COCO	71
CCP-1 Serial/Parallel Int.	BOTEK INSTRUMENTS	H	COCO	68
Disk Interface/ROM Pack Ext.	SPECTRUM PROJECTS	H	COCO	68
RS-232/C Expansion Cable	SPECTRUM PROJECTS	H	COCO	66
Spectrum Stick	SPECTRUM PROJECTS	H	COCO	63
TRS-80 Model 100 Port. Comp.	TANDY CORPORATION	H	OTHER	67
Interpod	OXFORD COMP. SYS.	H	64/20	68
Smart Ascii	MIDWEST MICRO ASSOC	H/C	64/20	67
Robographics CAD-1	ROBO GRAPHICS	H/S	APPLE	67
VoiceBox II	THE ALIEN GROUP	H/S	ATARI	68
TYMAC Universal Tape Interface	MICRO-WARE DIST.	H/U	64/20	66
Vanilla PILOT	COMPUTER MARKETING	L	64/20	65
DataFax	LINK SYSTEMS	S	APPLE	70
Aztec C	MANX SOFTWARE SYS.	S	APPLE	70
ANA-List	SYNOPTIC SOFTWARE	S	APPLE	70
Micro Illustrator	KOALA TECHNOLOGIES	S	APPLE	68
Money Tool	HOWARD W. SAMS	S	APPLE	68
Modula-2	VOLITION SYSTEMS	S	APPLE	66
The Stripper	EIGEN SYSTEMS	S	COCO	68
NEWTALK	STAR KITS	S	COCO	66
The World of Counting	EDUCOMP ENTERPRISES	T	APPLE	71
Cdex Training for VisiCalc	CDEX CORPORATION	T	APPLE	67
The Visible Computer: 6502	SOFTWARE MASTERS	T	APPLE	66
Discover BASIC	STERLING SWIFT PUBL.	T	APPLE	64
CoCo	ISA SOFTWARE	T	C64	65
Disk Library	MODULAR MEDIA	U	APPLE	64
Insta-Load	EDEN II COMPUTING	U	APPLE	64
The Prime Plotter	PRIMESOFT CORP	U	APPLE	63
Apple Mechanic	BEAGLE BROTHERS	U	APPLE	61
BASIC Commander	MMG MICRO SOFTWARE	U	ATARI	63
Pro-Color-File	DERRINGER	U	COCO	71
64K Disk Utility Package	SPECTRUM PROJECTS	U	COCO	67
Disk Utilities with Repair	COMPUTERWARE	U	COCO	62
Bank Street Writer	BRODERBUND SOFTWARE	W	AP/AT	65
Super Text	MUSE	W	C64	71
Casual Writer	E.N. PUBLICATIONS	W	VIC	66

MICRO Program Listing Conventions

Commodore

LISTING	C64 KEYBOARD
Commands	
{CLEAR}	^ CLR
{HOME}	HOME
{INSERT}	^ INST
{DOWN}	CRSR DOWN
{UP}	^ CRSR UP
{RIGHT}	CRSR RIGHT
{LEFT}	^ CRSR LEFT

Colors

{BLACK}	CTRL 1 BLK
{WHITE}	CTRL 2 WHT
{RED}	CTRL 3 RED
{CYN}	CTRL 4 CYN
{PURPLE}	CTRL 5 PUR
{GREEN}	CTRL 6 GRN
{BLUE}	CTRL 7 BLU
{YELLOW}	CTRL 8 YEL
{RVS}	CTRL 9 RVS ON
{RVSOFF}	CTRL 0 RVS OFF
{ORANGE}	= 1
{BROWN}	= 2
{GREY 1}	= 3
{GREY 1}	= 4
{GREY 2}	= 5
{LT GREEN}	= 6
{LT BLUE}	= 7
{GREY 3}	= 8

Functions

{F1}	f1
{F2}	^ f2
{F3}	f3
{F4}	^ f4
{F5}	f5
{F6}	^ f6
{F7}	f7
{F8}	^ f8

Special Characters

{PI}	π ^ Pi Char
{POUND}	£ Pound Sign
{UP ARROW}	↑ Up Arrow
{BACK ARROW}	← Back Arrow

Atari

Conventions used in ATARI Listings.

Normal Alphanumeric appear as UPPER CASE:
SAMPLE
Reversed Alphanumeric appear as lower case:
yES (y is reversed)
Special Control Characters in quotes appear as:
{command} as follows:

Listing	Command	ATARI Keys
{UP}	Cursor Up	↑ ESC/CTRL -
{DOWN}	Cursor Down	↓ ESC/CTRL =
{LEFT}	Cursor Left	← ESC/CTRL +
{RIGHT}	Cursor Right	→ ESC/CTRL *
{CLEAR}	Clear Screen	ESC/CLEAR
{BACK}	Back Space	ESC/BACK S
{TAB}	Cursor to Tab	ESC/TAB
{DELETE LINE}	Delete Line	ESC/SHIFT DELETE
{INSERT LINE}	Insert Line	ESC/SHIFT INSERT
{CLEAR TAB}	Clear Tab Stop	ESC/CTRL TAB
{SET TAB}	Set Tab Stop	ESC/SHIFT TAB
{BEEP}	Beep Speaker	ESC/CTRL 2
{DELETE}	Delete Char.	ESC/CTRL BACK S
{INSERT}	Insert Char.	ESC/CTRL INSERT
{CTRL A}	Graphic Char.	CTRL A
	where A is any Graphic Letter Key	

Non-Keyboard Commands

{DIS=}	CHR$(8)
{ENB=}	CHR$(9)
{LOWER CASE}	CHR$(14)
{UPPER CASE}	CHR$(142)
{^RETURN}	CHR$(142)
{DEL}	CHR$(20)
{SPACE}	CHR$(160)

Notes:

1. ^ represents SHIFT KEY
2. = represents Commodore key in lower left corner of keyboard
3. CTRL represents CTRL key
4. Graphics characters represented in Listing by keystrokes required to generate the character
5. A number directly after a {SYMBOL} indicates multiples of the SYMBOL: {DOWN6} would mean DOWN 6 times

Advertiser's Index

Coming in June

As a special bonus to Micro readers, we are including the complete all-new Apple IIe Supplement to What's Where in th Apple.

We will also share with you the fruits of someone's seven years of labor ... a Random Number Generator that has endless possibilities. For those unfamiliar with Macro's, we have an informative article explaining what Macro's are and how to incorporate them in your programming. The musical minded will enjoy our Musical Notes article putting a 5-octive range at your fingertips.

WHERE'S THE MicroCalc !

Those of you who took advantage of our recent subscription promotion which featured a free copy of our new **MicroCalc** Screen-Oriented Calculation Program - please be patient a little while longer. Our original plan was to make a few 'minor' improvements to the MicroCalc that was published in MICRO 68 (December 1983) and release it on disk. Well, once we got into making changes, we sort of got 'carried away'. This has been the primary cause of the delay.

The version of MicroCalc to be released shortly has many major improvements and completely new functions. These include:

☐ the ability to handle strings and string functions as well as numbers,
☐ program control functions for looping and testing limits,
☐ informative help screens,
☐ disk I/O routines that allow for automatic calling of subsidary screens from disk,
☐ printer routines for dumping the display screen,
☐ printer routines for generating formatted output,
☐ plus an extensive manual, complete listings, and demonstration screens.

Due to these additional features, and the extra effort that has gone into development of the MicroCalc package, the price has been increased from $14.95 to $29.95. Those of you who have already ordered MicroCalc, or who are owed it as part of your subscription, will not be charged anything extra.

We are sorry it has taken the extra time, but you will find that the time was well spent. The Commodore 64 version will be completed by the time you read this. The Apple version will be available in May and the Atari and CoCo versions in May or June.

Disk Service Now Available.

In response to your requests, we are now offering selected programs from recent issues on diskette. We will expand this service, if there is adequate demand. Each diskette will include all of the programs in BASIC and/or Assembly Source, plus binary 'load-and-go' files. The price includes shipping and handling.

Master Disk Directory
Charles Hill
Apple II with disk
MD-1 MICRO 67/69

Does-it Monitor
Michael Keryan
Commodore 64
MD-2 MICRO 68/69/70/71

Accurate Printer
Richard Marmon
Atari with Epson Printer
MD-3 MICRO 71

Send us your requests. If there is enough interest for any particular program, we will issue a diskette.

Please order each diskette by Name and Number
Send $15.00 for each diskette to:

MICRO Diskettes
P.O. Box 6502
Chelmsford, MA 01824