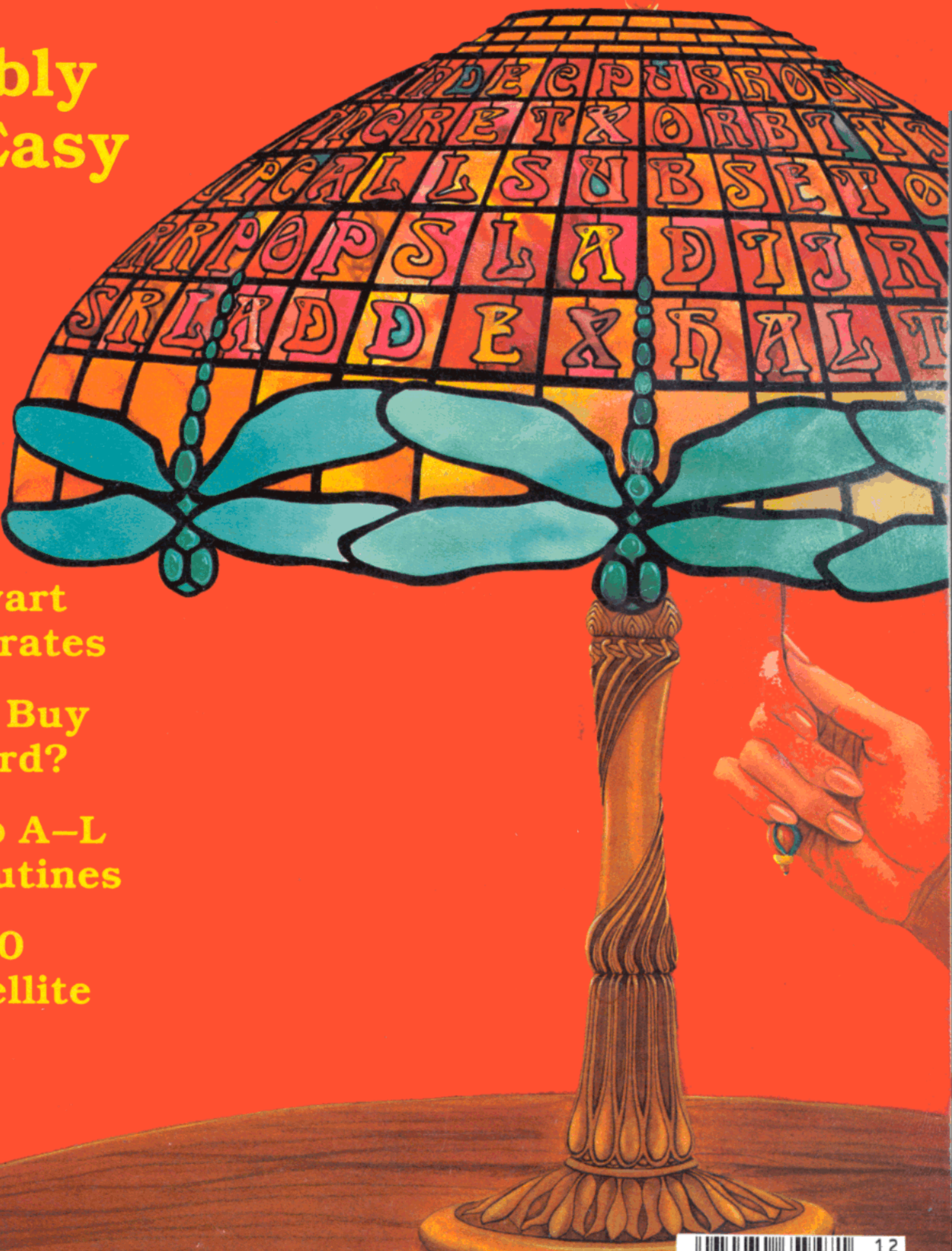
December 1983 USA $4.00

80micro

A WAYNE GREEN PUBLICATION

the magazine for TRS-80* users ®

Assembly Made Easy



Also inside:

How to Thwart Software Pirates

Should You Buy A CP/M Board?

The Road to A–L Disk I/O Routines

Your TRS-80 Thinks Satellite

The Fastest File-Search Program Around

12
0 74470 65947

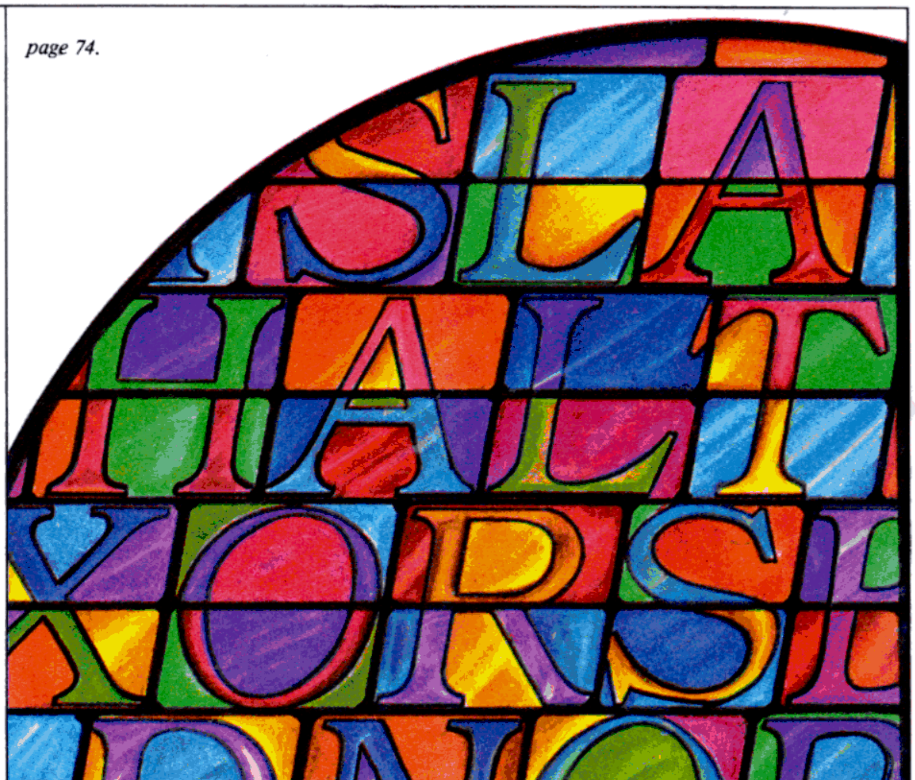
80 Contents

Features

Articles

Hardware

page 74.



Review

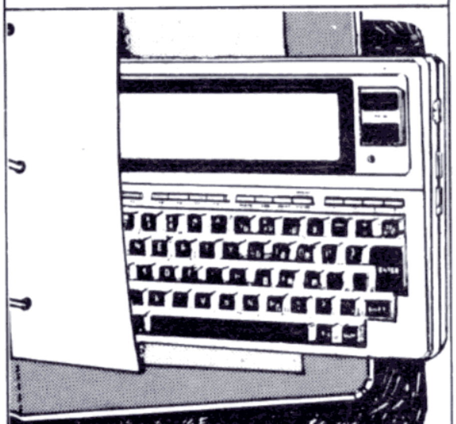
Tutorial

Utility

Model II/12/16

Departments

80 Contents



*TRS-80, Scripsit, and TRSDOS are trademarks of Tandy Corp.

PUBLISHER/PRESIDENT
Wayne Green
VICE PRESIDENT/GENERAL MANAGER
Debra Wetherbee
VICE PRESIDENT/FINANCE
Roger Murphy
ASSISTANT TO PRESIDENT/VP
Matt Smith
ASSISTANT TO VP/FINANCE
Dominique Smith
DIRECTOR OF MARKETING & SALES
David Schissler
DIRECTOR OF CIRCULATION
William P. Howard
603-924-9471
BULK & NEWSSTAND SALES MANAGER
Ginnie Boudrieau
1-800-343-0728
DIRECTOR OF ADVERTISING
Stephen Twombly
603-924-7138
Sales Manager: Edward Borezo
Sales: Betty Butler,
Judi Wimberly
Ad Coordinator: Mary Hartwell
PUBLIC RELATIONS
James Leonard

PRODUCTION
Director: Nancy Salmon
Assistant: David Wozmak
Michael Ford, Marjorie Gillies,
Kimberly Nadeau, Phyllis Pittet,
Anne Rocchio, Lynne Simonson,
Kenneth Sutcliffe, Karen Wozmak;
Film Production: Donna Hartwell,
Theresa Verville, Robert M. Villeneuve;
Ad Coordinators: Patricia Bradley,
Paula Ramsey;
Assistant: Taylor Morris;
Advertising Production: Jane Preston,
Fiona Davies, Bruce Hedin,
Scott Philbrick
PHOTOGRAPHY
Supervisor: Thomas Villeneuve;
Sandra Dukette, Nathaniel Haynes,
Laurie Jennison, Sturdy Thomas
TYPESETTING
Supervisor: Sara Bedell; Darlene Bailey, Prem Gongaju,
Lynn Haines, Cynthia Letourneau, Lindy Palmisano,
Heidi Thomas, Sue Weller
DESIGN
Supervisor: Joyce Pillarella;
Susan Donohoe, Holly Fuette,
Howard Happ, Beth Krommes,
Dion Owens, Dianne Ritson,
Patrice Scribner, Susan Stevens,
Suzanne Torsheya, Sarah Werninger,
Donna Wohlfarth;
Copywriters: Louis Marini, Gail Morrison, Dale Tietjen,
Steve Tripp
DESIGN DIRECTOR
Christine Destrempes

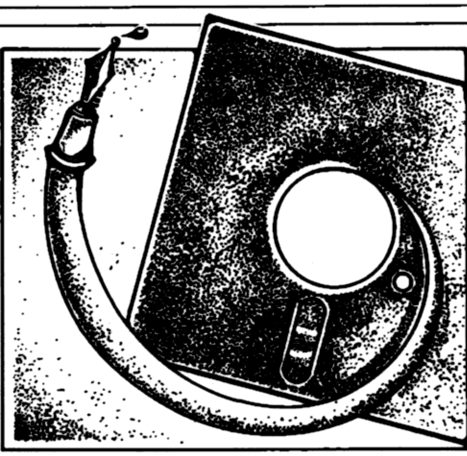
Cover by Damian Henriques and William Giese

80 Micro is a member of the CW Communications/Inc. group, the world's largest publisher of computer-related information. The group publishes 42 computer publications in 18 major countries. Nine million people read one or more of the group's publications each month. Members of the publication group include: Australia: *Australasian Computerworld, Micro Magazine*; Argentina: *Computerworld/Argentina*; Brazil: *DataNews, MicroMundo*; Denmark: *Computerworld/Danmark, MikroData*; France: *Le Monde Informatique*; Germany: *ComputerWoche, MicroComputerWelt, PC Welt*; Italy: *Computerworld Italia*; Japan: *Computerworld Japan*; Mexico: *Computerworld/Mexico*; Norway: *Computerworld Norge, MikroData*; People's Republic of China: *China Computerworld*; Saudi Arabia: *Saudi Computerworld*; Spain: *Computerworld/Espana, MicroSistemas*; Sweden: *ComputerSweden, MikroDatorn, Min Hemdator*; United Kingdom: *Computer Management, Computer Business Europe*; United States: *Computerworld, Desktop Computing, HOT CoCo, inCider, InfoWorld, ISO World, Microcomputing, PC World, 80 Micro, RUN*.

80 Micro (ISSN 0744-7868) is published 12 times a year by 1001001 Inc., 80 Pine St., Peterborough, NH 03458. Phone: 603-924-9471. Second class postage paid at Peterborough, NH, and additional mailing offices. Subscription rates in U.S. are $36 for one year, $55 for two years, and $75 for three years. In Canada and Mexico $45—one year only, U.S. funds drawn on a U.S. bank. Canadian distributor: Micron Distributors, 409 Queen St. West, Toronto, Ontario, Canada M5V 2A5. BC Canadian Distributor: Compulit, 1122 Fir St., Blaine, WA 98230. Foreign subscriptions (surface mail), $55—one year only, U.S. funds drawn on a U.S. bank. Foreign subscriptions (air mail), please inquire. In South Africa contact *80 Micro*, P.O. Box 782815, Sandton, South Africa 2146. All subscription correspondence should be addressed to *80 Micro*, Subscription Department, P.O. Box 981, Farmingdale, NY 11737. Please include your address label with any correspondence. Postmaster: Send form 3579 to *80 Micro*, Subscription Services, P.O. Box 981, Farmingdale, NY 11737.

Editorial:
Send all correspondence to *80 Micro*, Pine St., Peterborough, NH 03458.
Subscriptions:
Problems with Subscriptions: Send a description of the problem and your current and/or most recent address to: *80 Micro*, Subscription Department, P.O. Box 981, Farmingdale, NY 11737.
Problems with Load 80 Circulation: Address correspondence to Lori Eaton, 80 Pine St., Peterborough, NH 03458.
Change of Address: Send old label or copy of old address and new address to: *80 Micro*, P.O. Box 981, Farmingdale, NY 11737. Please give eight weeks advance notice.
Microfilm: This publication is available in microform from University Microfilms International. United States address: 300 North Zeeb Road, Dept. P.R., Ann Arbor, MI 48106. Foreign address: 18 Bedford Row, Dept. P.R., London, WC1R4EJ, England.
Dealers: Contact Ginnie Boudrieau, Bulk Sales Manager, *80 Micro*, Pine St., Peterborough, NH 03458. (800) 343-0728.

by Eric Maloney



America's happiest revolution

We've already seen one so-called revolution fail. It evaporated into thin air shortly after Kent State, unable to bear the weight of its own rhetoric. So pardon me if I'm skeptical about this latest revolution supposedly being fomented by the personal computer.

At least, the media and self-styled experts on social change are calling it a revolution. If so, it's got to be the happiest one in the history of the world. It sparkles beneath a California sun that shines like a big yellow smile button. It glows brightly from the pages of *Newsweek* and *The New York Times*, from quarter-million dollar television commercials, from the zippy pop-computer magazines that hang fat and lazy from the newsstands.

Here is one revolution that promises to do you no harm. Nobody gets hurt, nobody gets agitated. With the exception of a few malcontents, nobody even gets upset.

I can't claim to be an expert on revolutions, since I've never been in one. But it's my understanding that a revolution is aimed at the fundamental concepts upon which a political or social structure is built. Take the Industrial Revolution, for example. It caused social and economic upheaval on a massive scale. It shifted power from the land-owners to the factory owners, from the rural areas to the cities, and completely changed the way people lived and worked.

But the microcomputer revolution is another story. It does nothing to challenge the basic relationship between people and their supporting institutions. It does not change the make-up of the ruling class, nor does it alter our perceptions of that class.

Some people say that microcomputers will give us more control over our lives, that computers will let us access information heretofore available only to the rich and powerful. This may, to some extent, be true; if information is power, greater access to information means greater power. But if the average citizen gains a little, the ruling class will gain a good deal more.

Ultimately, computers will serve to reaffirm the social and political hierarchies in this country. The strong will be stronger and the weak weaker, a trend that hardly can be called revolutionary.

* * * * *

Adam Osborne has been one of the micro industry's most caustic critics. So more than a few people chortled when Osborne Computer Corp. collapsed last summer.

In some respects, Osborne comes out looking like a fool. When he introduced the Osborne, he let loose a barrage of criticism at his fellow manufacturers, predicting in the May 1981 issue of *Microcomputing* that "their unrealistic perspective will trigger disaster with all the tragic ruthlessness that early success had on such companies as Imsai and Processor Technology."

But Osborne's failure was not due to lack of insight. In retrospect, his comments in that article often hit the mark. "Hardware manufacturers must concentrate on driving down the price of hardware," he warned. And further on, he predicted "a new, massive round of hardware price reductions."

Osborne also saw the impact his new computer would have on the micro market, and predicted "a rapid evolution of new, low-cost portable microcomputers that appeal to individuals and are used with the frequency of typewriters." And, in a forecast fraught with irony, he saw that by 1983 the Osborne would "have a lot of company."

Finally, though IBM had not yet entered the arena, Osborne saw the eventual establishment of industry-standard software and hardware. "Those who stray from industry standards will be forced to leave the microcomputer marketplace," he said.

Osborne's sin was that he was as lousy a businessman as the early losers he criticized. And apparently, the major manufacturers were not as misguided as he thought they were. Ultimately, they changed their ways, learning to see and respond to shifts in the marketplace.

Osborne was an important stimulus to those changes. He may not have practiced what he preached, but he had a major impact on the industry and the way it sold itself. He should be given appropriate credit.

* * * * *

Last week at the Peterborough Diner, I gave the cashier $2 for a $1.95 turkey sandwich. The computerized cash register credited me with 9 cents' change. The waitress dutifully gave me a nickel and four pennies.

"Wait a minute," I said, and pointed out that $2 minus $1.95 was not 9 cents. She puzzled over the problem for a moment, finally saw the light, and took back the surplus change.

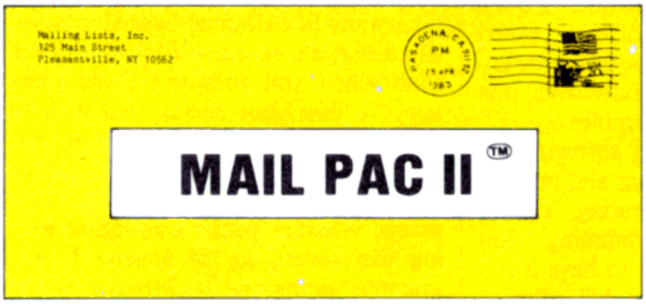
This little tale has several morals. First, we must retain our ability to do simple math.

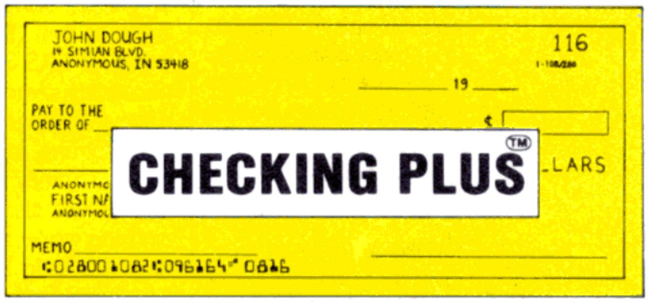
Second, we must not believe everything the computer says, or let it subvert our common sense.

And third, we must be watchful consumers. As long as human beings punch the buttons, the buyer's motto remains "caveat emptor." ■

ANYONE CAN USE OUR SOFTWARE!

MAIL PAC II and CHECKING PLUS represent a new generation of computer software. Our software doesn't even need a manual! Anyone can quickly learn to use either package by following simple, explicit on-screen instructions. But, for those of you who insist -- we've included a complete user's manual as well.





MAIL PAC II FEATURES:

High Capacity --

Stores from one to one million names in Zip Code, Numerical or Alphabetical order. The only limitation is your disk storage space.

Flexibility --

Prints your mailing list on 1, 2, 3, or 4-across labels (with up to 5 user-defined lines on each label) or as a compact, user-designed directory. The record length is completely user-defined, and each field within each of your records is completely variable (allowing storage of any number of characters for any particular address entry). Devote fields to telephone numbers, codes, or even special messages related to each particular name on file.

Powerful --

A built-in word processor allows you to create personalized form letters for each address on your list (or just a particular group of addresses).

Ease of Use --

Create new mailing lists, review existing lists, handle changes of address, delete cancelled names, sort lists, and purge duplicate names from your files. Complete on-screen instructions tell you in plain english exactly how to accomplish all of these tasks.

MAIL PAC II $99.95

CHECKING PLUS FEATURES:

Complete Check Register --

Checking Plus stores your entire check register in a disk file, and then uses the data to balance your account, track your expenses, and help you make budget projections. Review the entire checkbook, enter checks, deposits, fees and adjustments, mark outstanding checks when paid, and get an instant cash balance at any time. All data can be viewed onscreen or printed out in report form.

Tax Preparation --

Storage of monthly and yearly totals and other important information aids in income tax preparation, for your personal use or for your accountant.

Handles the Details --

Store names and addresses for frequently written checks, then print checks to fit standard window envelopes, eliminating the need for extra typing. You can even store any comment, explanation or other message (up to 255 characters) related to a particular check.

Automatic Monthly Bill Payments --

Enter amounts and names of payees for all of your fixed-cost monthly payments, and then sit back while the system automatically prints checks.

CHECKING PLUS $99.95

Both Programs Require 48K And At Least One Disk Drive (2 Drives For Higher Storage Capacities). And All Hard Disk Systems Are Supported.	**FREE**–TRS-80 Mod I,III & 4 programs supplied on DosPlus (minimum system). Complete DosPlus also available.

FOR YOUR TRS-80™ • APPLE™ • IBM PC™ • NEC™ • OSBORNE™ • XEROX™ • KAYPRO™ • TELEVIDEO™ • ZENITH™ • SANYO™
DEC™ • TI PROFESSIONAL COMPUTER™ • SUPERBRAIN JR.™ • EPSON™ • Any CP/M™ Computer

CP/M-based Computers must be equipped with Microsoft BASIC (MBASIC or BASIC-80)

OUR SOFTWARE CATALOG

H & E Computronics, Inc., has mailed more than 1 million software catalogs since 1978. Send $2 for our new 64-page catalog today! (We also send you our catalog FREE with every order). DEALER INQUIRIES WELCOME

30-DAY MONEY BACK GUARANTEE

*** ALL PRICES & SPECIFICATIONS SUBJECT TO CHANGE ***
DELIVERY SUBJECT TO AVAILABILITY

H & E COMPUTRONICS INC.
50 N. PASCACK RD., SPRING VALLEY, N.Y. 10977

24 HOUR ORDER LINE
(914) 425-1535

ADD $3.00 FOR SHIPPING IN UPS AREAS
ADD $4.00 FOR C.O.D. OR NON-UPS AREAS
ADD $5.00 TO CANADA & MEXICO
ADD PROPER POSTAGE OUTSIDE OF U.S., CANADA & MEXICO

NEW TOLL-FREE ORDER LINE
(OUTSIDE OF N.Y. STATE)
(800) 431-2818

VISA MasterCard

↙9

The left bracket, [, replaces the up arrow used by Radio Shack to indicate exponentiation on our printouts. When entering programs published in *80 Micro*, you should make this change.

***80* formats its program listings to run 64-characters wide, the way they look on your video screen. This accounts for the occasional wrap-around you will notice in our program listings. Don't let it throw you, particularly when entering assembly listings.**

Article submissions from our readers are welcomed and encouraged. Inquiries should be addressed to: Submissions Editor, 80 Pine Street, Peterborough, NH 03458. Include an SASE for a copy of our writers' guidelines. Payment for accepted articles is made at a rate of approximately $50 per printed page; all rights are purchased. Authors of reviews should contact the Review Editor, 80 Pine Street, Peterborough, NH 03458.

PROOF NOTES

The editors look at the issues

Speaking in hex

You asked for it, though in a roundabout fashion and probably without your knowing it. *80 Micro*'s readership surveys tell us a lot about your needs. One thing we've noticed is your growing interest in Assembly language. This isn't too surprising since most of you (88.5 percent to be exact) use your TRS-80 for home or hobby applications. And we all know what that means: software programming.

Designing and writing software, especially games, is both fun and rewarding. It can also be frustrating. There's nothing worse than programming a fantastic game in Basic only to have it run as slow as molasses uphill. Screen changes take seconds and your rockets never quite seem to reach their targets in time. This is usually when your interest in Assembly language begins to take root and grow. This issue will nurture that growth.

Communicating with your computer in a high-level language like Basic is like talking with someone who speaks a foreign language. When you say goodbye to a Frenchman, he has to use a dictionary to see that you mean *au revoir*. The computer follows the same process. If you write a Basic instruction in your program, the computer has to translate this instruction to its "native tongue," machine code.

When you use a low-level language like Assembly language, the interpretive process is greatly simplified. The computer operates on Assembly-language instructions much faster because that language is closer to the Z80's machine code.

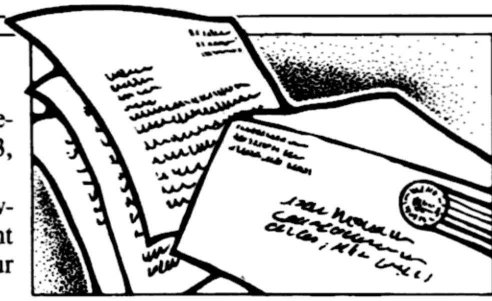
Assembly-language programs execute up to 300 times faster than Basic. They also require less memory: you can run an Assembly-language program in 4K that normally requires 24K in Basic. And if you ever had an urge to see how the Z80 processes all of those routines in ROM or TRSDOS, Assembly language lets you do that as well.

Assembly language can be intimidating at first with its extensive use of binary and hexadecimal data. But be assured that it's no more difficult to learn than when you mastered Basic years ago. To ease your nerves, and to help build your confidence, we've put together a collection of articles that cover the entire spectrum of Assembly language. Whether you're a neophyte testing the waters, or an advanced programmer doing the backstroke, you'll find something to meet your needs between this month's covers.

For example, Hardin Brothers presents his first installment of *Assembly Language Made Simple* for this issue. In this comprehensive piece he gently introduces you to your first dose of Assembly language. And since everyone loves to learn shortcuts, Bob Bowker continues to share some of his secrets with us. You'll find them in part two of his article, *Assembly-Language Shortcuts* (p.). They are especially useful for those of you who've been introduced to the language.

Other articles of interest include David Haan's technique titled *Assembly Language Disk I/O*, Terry Kepner's feature review, *CP/M III Ways*, and Joseph Trojak's *Finding the Search Solution*, a program that lets you search text files quickly. And when the day ends, you can relax in your easy chair and watch one of over 140 TV channels, made available with help from Dan Keen and Dave Dischert's article, *Channels of Communication*. If you don't want to sit in front of the boob tube all night, there's always the antics of the Gamer's Cafe to keep you smiling. So sit back and enjoy! ■

—S.F.T.
01010000



MaxiStat Now StatPac

John Harrell did an excellent job reviewing MaxiStat (September 1983, p. 50).

Your readers should be aware, however, that we have made important changes relative to MaxiStat since your review appeared.

MaxiStat is no longer published or marketed by The Business Division. Walonick Associates now markets and supports an updated version of the package under its original name of StatPac.

This update contains substantial changes from the version you reviewed, including more statistics, printing options, and machine-language subroutines for speed.

Anyone interested in further information should contact Walonick Associates.

David S. Walonick
President, Walonick Associates
5624 Girard Ave. South
Minneapolis, MN 55419

Printer Fix

The DMP200 printer has one drawback: Its superscripts are low, and the rest of the text after a subscript is noticeably lower than the text preceding the subscript.

To rectify this, change the half-reverse line feed action to a full-reverse line feed followed by a half-forward line feed (see Program Listing 1). This has an insignificant effect on print speed, and it works.

This fix also works on the Radio Shack Line Printer VIII, and with other Radio Shack printers that use the same printer drivers.

To install this modification, define the user print code as the control-@ key sequence, since the @ key is close to the * key. Use the System Set-up option from the main menu and then the Enter Printer Codes option.

Under the @ key, enter 0 for the units and 27,10,27,28 for the sequence of codes.

You can also change printer driver DMP400/CTL or LP8/CTL by replacing the half-reverse line feed code with the code for a full-reverse line feed followed by a half-forward line feed (second half of Listing 1). This lets you use the control-* code exactly as normal, and there is no interference with any observable function of SuperScripsit 1.1.

You must type in the patches exactly as shown below, and only to your backup copies of the program (building a do-file is a very convenient way to save on repeated typing).

Ian McCauley
Department of Biochemistry
University of Western Australia
Nedlands, 6009


```
PATCH DMP400/CTL (ADD=BCAC,FIND=1C,CHG=0A)
PATCH DMP400/CTL (ADD=BCB0,FIND=C9,CHG=00)
PATCH DMP400/CTL (ADD=BCB8,FIND=1E,CHG=1C)
PATCH DMP400/CTL (ADD=BB4D,FIND=A5,CHG=B1)
PATCH DMP400/CTL (ADD=BB50,FIND=B1,CHG=A5)
PATCH DMP400/CTL (ADD=BC76,FIND=B1,CHG=A5)
PATCH DMP400/CTL (ADD=BC82,FIND=A5,CHG=B1)
                      OR
PATCH LP8/CTL (ADD=BC9E,FIND=1C,CHG=0A)
PATCH LP8/CTL (ADD=BCA2,FIND=C9,CHG=00)
PATCH LP8/CTL (ADD=BCAA,FIND=1E,CHG=1C)
PATCH LP8/CTL (ADD=BB4D,FIND=97,CHG=A3)
PATCH LP8/CTL (ADD=BB50,FIND=A3,CHG=97)
PATCH LP8/CTL (ADD=BC68,FIND=97,CHG=A3)
PATCH LP8/CTL (ADD=BC74,FIND=97,CHG=A3)
```

Program Listing 1. Patch for the DMP200 printer.

LPVIII Expanded Graphics

In Jim Hanson's letter to Input (June 1983, p. 12), his program to display screen graphics to the LPVIII prints graphics horizontally shortened by one third.

Program Listing 2 is a modification to Hanson's program that solves this problem. Lines 900 and 910 define the block and space print location. Line 920 defines the tab (LPVIII POS command) to center the display on the page.

Don Moore
P.O. Box 1405
Coconut Grove, FL 33133


```
900 BK$ = CHR$(255)+CHR$(255)+CHR$(255)
910 SP$ = CHR$(128)+CHR$(128)+CHR$(128)
920 TB$ = CHR$(27)+CHR$(16)+CHR$(0)+CHR$(48)
1000 LPRINT CHR$(18);
1010 FOR H = 0 TO 47
1015 LPRINT TB$;
1020 FOR W = 0 TO 127
1030 IF POINT  (W,H) THEN LPRINT BK$; ELSE LPRINT SP$;
1040 NEXT W
1050 LPRINT ""
1060 NEXT H
```

Program Listing 2. LPVIII print adjustment.

Machine-Specific Programs

I am distressed by the concept, stressed throughout *80 Micro*, that every program published must be compatible with both the Model I and Model III.

Most of the programs in your magazine that run on the Model III also run on a Model I, and it's the same story for many software companies.

The reasons for this phenomenon are simple and well intentioned: Having the same program work on both machines satisfies twice as many people, means

Error Trap

In our "Buyer's Guide to Disk Drives and Disks" (June 1983, p. 234), we omitted Micro Design. Micro Design (6301 Manchaca Road, Austin, TX 78745, 800-531-5002) markets one 40-track, 5¼-inch disk drive for the Model I, and three systems for the Model III.

Micro Design offers 40-track, single-sided 5¼-inch disk drives with 250K storage capacity (first upgrade is $399, second internal drive is $189.95). Also available are 40-track, double-sided disk drives with 500K storage capacity (first drive upgrade is $474, second drive is $274), and 40-track, 5¼-inch, double-sided drives with 500K storage capacity (disk drive upgrade is $495, additional slim line drive is $295).

In the September Input column (p. 12), we omitted the program listing for Howard Potvin's letter. Program Listing 6 eliminates the extra alpha column headings in John Corbani's Minicalc program (May 1983, p. 140).

Line 470 contains the code to set an Epson printer to 132 characters per inch, and Potvin changed the reference to the number of columns in line 110 from 14 to 15. Line 475 sets unidirectional printing and checks for number of columns.

If you have eight or fewer columns, set Q to 1 and drop to line 480. If you have more than eight, set Q to 9 and go to line 476 to print the first eight columns. Then go to line 480 and print the rest. Line 49 keeps the underline length correct.

These changes print the full 16 columns by 26 rows on one sheet of standard sized paper.

Richard Green's review of Power DOT (October 1983, p. 80) contains a minor error. On page 84, the two angles referred to as 150 degrees are actually 15 degrees.

you have to write only one version of the program, and guarantees that you can keep the software if you upgrade from a Model I to a Model III.

I cannot help thinking, however, that this practice is short-changing owners of Model IIIs and modified Model I's. These programs don't let me use my special features at all.

As an example, Radio Shack obviously designed its Series I Editor/Assembler for a Model I. Only the left shift key works, and I must use the low cassette speed. To generate lowercase characters, I have to depress the shift key.

Ideally you should have two versions of the same program, each designed for its own individual machine. An excellent example of this is Frolic, the machine-language monitor contained in the book *TRS-80—More Than Basic*, by John Froehlich.

Mr. Froehlich wrote two different monitor programs, one for each model, each version using its own model's special features. Why can't major software manufacturers do this as well?

I urge the editors of *80 Micro* to encourage machine-specific program submissions. Once you have such programs, you could convert them, much the same as you do in the Take II column.

I have used both the Model I and the Model III, and it isn't difficult to alter existing generic programs to fully utilize each machine's specialties.

Once people stop bunching together the Model I and III as one machine, the owners of both will be better off. However, the trend does not seem to be in that direction.

A serious flaw in Model 4 Basic is the necessity of separating each Basic key word with a space. Will programs published in the future in magazines and on disk and tape have spaces between each keyword, to make every program compatible with that machine too? I sincerely hope not.

Mark Allen Reed
Box 459-A
Lyme Center, NH 03769

AIDS-III Space Saver

Although the AIDS-III packages published in the March and April 1983 issues (p. 136 and p. 168, respectively) of *80 Micro* are the most useful to be found, they take up a lot of disk space.

I've come up with a way to save a great deal of disk space and keep several files on one disk.

Make up individual data lists (all starting at line 5000) and save them as ASCII files (use the ,A option). Then make up a menu for the programs you've developed (see Program Listing 3), plug the menu into lines 1-7 of the AIDS package, and you can use the same basic AIDS program to load any number of data files.

After you type in RUN, the program asks if you've loaded a program yet. Answer anything but Y. The program then brings up the menu. By selecting the number of the desired data file, you can merge it with the original AIDS package. At the ready prompt, type RUN, answer Y this time, and press the enter key.

Using this procedure, I'm able to get AIDS-III, MAPS-III, MERGE-III, CALCS-III, and eight data files (saved in ASCII), their descriptor files, and their record files on one disk.

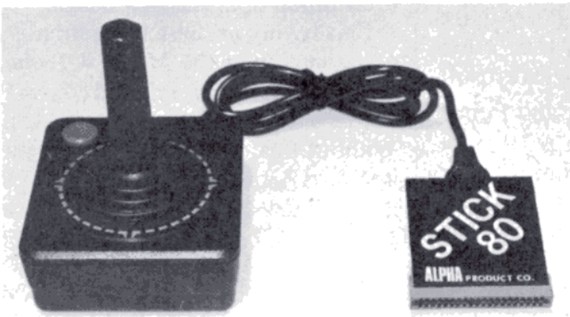
```
1 CLS:PRINT@596, "HAS A PROGRAM BEEN LOADED YET?": INPUT A$: IF
A$="Y" THEN 10 ELSE 3
3 CLS:PRINT@74, "THE FOLLOWING PROGRAMS ARE DEVELOPED:": PRINT@2
02,"1. VIDEO LISTINGS.": PRINT@266,"2. FOOD CABINET INVENTORY.":
PRINT@330,"3. FREEZER INVENTORY.":PRINT@394,"4. FRIENDLY PARTIES
.":PRINT@458,"5. HOUSEHOLD INVENTORY."
4 PRINT@522,"6. SUBORDERS.": PRINT@586,"7. UNDEFINED.": PRINT@65
0,"8. EXIT PROGRAM.": PRINT@842,"SELECT CHOICE...": INPUTN: IFN<
1ORN>8 THEN CLS: GOTO 3 ELSE CLS
5 IFN=1 THENPRINT@596,"MERGING VIDEO PROGRAM": MERGE "VIDEODES/F
IL" ELSEIFN=2 THEN PRINT@596, "MERGING FOOD CABINET PROGRAM": ME
RGE "FOODDES/FIL" ELSEIFN=3 THEN PRINT@596, "MERGING FREEZER PRO
GRAM": MERGE"FREEZDES/FIL" ELSE6.
6 IF N=4 THENPRINT@596,"MERGING FRIENDLY PROGRAM": MERGE"FRENDDE
S/FIL" ELSEIFN=5 THEN PRINT@596,"MERGING INVENTORY PROGRAM": MER
GE "HOUSEDES/FIL" ELSEIFN=6 THENPRINT@596,"MERGING SUBORDER PROG
RAM": MERGE "SUBSDES/FIL" ELSE7
7 IF N=8 THEN STOP: ENDELSEIF N>8ORN<8 THENCLS: FORI=1TO100: PRI
NT@596,"UNDEFINED...":NEXTI:GOTO3
```

Program Listing 3. Menu for the AIDS-III program.

THE ALPHA JOYSTICK -30%

NOW ONLY $28^{00}

SPECIFY MOD I OR III



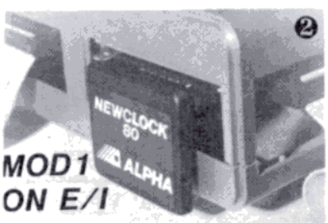
DON'T WAIT!

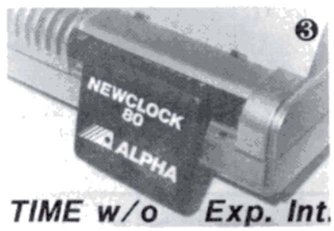
Limited time special

Works with all joystick games.

Model IV: order model III version

$59.95 NEWCLOCK-80





Wouldn't it be nice if your computer could always boot up with the right time and date and then stay accurate. Newclock-80 will enhance your Model I or III system with powerful clock/calendar/timer functions.

Using LSI (large scale integration) and custom circuits, Newclock-80 provides MO/DATE/YR, HR:MN:SEC plus AM/PM and day of week and even takes care of leap years! It continues to keep time and date with quartz accuracy when the computer is turned off or experiences a power failure. A single battery lasts over 2 years.

Compatibility: Newclock-80 is compatible with any operating system, including DOSPLUS, NEWDOS, LDOS. With its fully decoded circuitry it will work with any other hardware you may own. Bus expanders are available.

Installation is very simple, no tools, no disassembly, no soldering. Just plug it in, that's all. There is no power supply or messy cable. Newclock-80 plugs into the rear of the keyboard ❸ or side of the Exp. Int. ❷. Model III Newclock fits the 50 pin card edge (underneath) ❶

The Software: Newclock-80 is as easy to use as it is to install. -"SET", a Basic program, is used only once to set the time and date and select 12 or 24 hour format. -"TIMESTR", also in Basic, patches your computer "TIME$" function to read Newclock-80. It also adds "TIME$" to keyboard-only systems, a short routine is simply "poked" into low memory.

Newclock-80 uses 12 ports (176 to 188): 6 for the time, 6 for the date. The data is conveniently stored in decimal form, no conversion is needed. You can read or modify any digit using simple Basic "INP" and "OUT" statements.

No risk trial. Order your Newclock-80 today, see how easy it is to install and operate then decide within 30 days if you want to keep it. If for any reason you are not delighted with its quality and performance, you may return it for a prompt and courteous refund.

Your unit will come complete ❹ with software on tape, detailed instructions, handy reference card, and a 90 day warranty. Specify Model I or III. Software is also available on disk: add $5.
Lithium battery (not included) available from RADIO-SHACK (#23-162) or add $1.50 to your order.

Thanks to outstanding engineering and efficient manufacturing, ALPHA Products is once again able to offer a great product at a surprising price. Order your Newclock-80 at no obligation today.

Toll Free Order Line
800·221·0916
Orders Only. NY & Info call (212) 296-5916. Hours: 9-5 E.S.T.

ALPHA Products

79-04 Jamaica Ave., Woodhaven, NY 11421 ✓17 (212) 296-5916

ADD $2.50 PER ORDER FOR SHIPPING AND HANDLING.
WE ACCEPT VISA, MASTERCARD, CHECKS, M.O.
C.O.D. ADD $3.00 EXTRA.
NY RESIDENTS ADD SALES TAX.
OVERSEAS, FPO, APO: ADD 10%.
DEALER DISCOUNTS AVAILABLE.

Now if someone can show me a way to get Basic to run after a merge without typing in RUN, I'd really be happy!

James T. Krywalski
80 Lockwood Ave.
Buffalo, NY 14220

AIDS-III Error

An error exists in the lowercase support program modifications that appeared in the August 1983 issue of *80 Micro* (p. 12). The modifications support the lowercase characters, but disable the up-arrow (K1) key so it no longer functions to move the cursor to the previous entry lines.

I corrected this by adding and changing the following program lines to the AIDS-III program, in addition to making the changes that appeared in the August 1983 issue.

```
170 IFIC=91THENIC=6
2000 US=CHR$(91): CC$=U$+ CHR$(K2)+
    CHR$(K5)+ CHR$(K6)+ CHR$(K9)
3744 DATA 13,6,10,8,9,27,26,24,25,31,3
```

I use 123 instead of 6 in lines 170 and 3744, but suggest the use of 6. It's shorter and appears to work, although I haven't tested it thoroughly.

Also, line 640 of the August modifications for AIDS-III has a slight error in the last statement. SV = FV$ should read SV$ = FV$ to conform with the original program and the MAPS-III and CALCS-III changes.

Darrell Lee
The Food and Drug Administration
c/o Administration Building,
Room 1033
1745 W. 1700 South St.
Salt Lake City, UT 84104

Maze Chase PEEK

The changes in Program Listing 4 are for Leonard Karr's Maze Chase game (August 1983, p. 272). They let you hold one key down and continue to move until you release the key.

Mike Cessna
610 N. 54th St.
Springfield, OR 97477

High-Speed POKE

For the Model 4 owner, Model III software operates at the Model 4's higher speed by utilizing the following POKE from Basic:

```
POKE 16912,104 (to speed up)
POKE 16912,40 (to slow down)
```

When assembled, the source code in Program Listing 5 permits toggling the speed-up off and on from the DOS READY command line.

R. L. Stuart
15434 Sherman Way
Van Nuys, CA 91406

More on the Model 4

Regarding the Model 4 review, "Once More, with Feeling" (August 1983, p. 100), I've found that other commands not supported by Microsoft Basic 5.0 include Set, Reset, and Point, limiting graphics production.

Also, space delimiters required after reserved words in Microsoft Basic 5.0 don't take up additional memory. The program removes these spaces when it converts Basic commands to single-byte tokens, and they are restored at print or list time.

It is true that the conversion utility (CONV) lets you copy files from TRSDOS 1.2 and 1.3 formatted disks to TRSDOS 6.0 formatted disks. However, this conversion is of minimal help in converting Model III Basic to Microsoft Basic 5.0 because mapping Basic commands to single-byte tokens is entirely different in the two Basics.

Finally, my greatest frustration is the unavailability of the *Model 4 Technical Reference Manual* referred to in the *Model 4 Disk System Owner's Manual.* Without this manual, it's impossible to interface Assembly language with either the Model 4 hardware, or with TRSDOS 6.0.

Gregory E. Nutt
23 Pendleton Lane
Londonderry, NH 03053

```
2100 L=PEEK(14400)
2110 IF L=8ANDPEEK(S+PL-64)...
2120 IF L=32ANDPEEK(S+PL-3)...
2130 IF L=16ANDPEEK(S+PL+64)...
2140 IF L=64ANDPEEK(S+PL+3)...
3070 PRINT"     UP ARROW MOVES YOU UP"
3080 PRINT"     DOWN ARROW MOVES YOU DOWN"
3090 PRINT"     LEFT ARROW MOVES YOU LEFT"
3100 PRINT"     RIGHT ARROW MOVES YOU RIGHT"
```

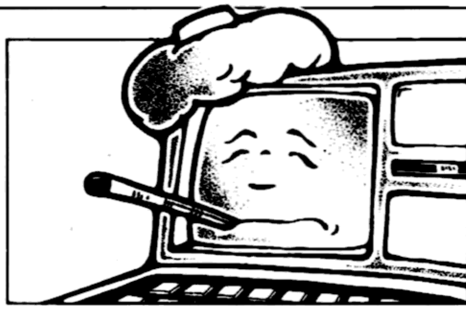
Program Listing 4. PEEK for Maze Chase.

```
;         ****************************
;         *         SPEED/SRC        *
;         * Speed Toggling Program   *
;         * For Model III Software   *
;         * Used on TRS-80 Model 4   *
;         ****************************
;
          ORG     0FF00H           ;PROGRAM LOAD ADDRESS
          LD      HL,16912         ;HL POINTS TO SPEED ADDRSS
          LD      A,(16912)        ;A=VALUE IN SPEED ADDRESS
          CP      104              ;IS IT 104? (HIGH)?
          JP      Z,LP1            ;IF HIGH, JUMP LP1 FOR LOW
          LD      (HL),104         ;IF LOW, MAKE HIGH
          JP      LP2              ;AND JUMP LP2 TO EXIT PGM.
LP1       LD      (HL),40          ;IF HIGH,MAKE LOW (40)
LP2       JP      402DH            ;JUMP TO DOS READY
          END     0FF00H
```

Program Listing 5. Speedup toggling program for the Model 4.

```
470 IFPEEK(14312)<>61THEN PRINT@960, "PRINTER NOT READY";:GOTO 470
 ELSE LPRINT CHR$(15)TAB(10) TY$:LPRINT
475 LPRINT CHR$(27) "U":IFMV<8THENQ=1:GOTO480 ELSE Q=9
476 LPRINT TAB(5);: FORR=1 TO 8: LPRINT "          "CHR$(R+64) "
 ";: NEXT: LPRINT: LPRINT: FORA5=1 TO MH+1: LPRINT TAB(5) A5TAB(9)
;: FORR=1 TO 8:T$=MID$(A$(A5),R*15-13,14): GOSUB 510: NEXT: LPRINT
477 IF A5=MHLPRINT TAB(5) STRING$(120,"-")
478 NEXT:  LPRINT: LPRINT: LPRINT
480 LPRINTTAB(5);: FOR R=Q TO MV+1: LPRINT "   "CHR$(R+64) "    ";
: NEXT: LPRINT: LPRINT: FOR A5=1 TO MH+1: LPRINTTAB(5
) A5 TAB(9);: FOR R=Q TO MV+1: T$=MID$(A$(A5),R*15-13,14): GOSUB 5
10: NEXT: LPRINT
490 IF A5=MHL PRINTTAB(6) STRING$(15*(MV-Q+2),"-")
500 NEXT: LPRINT: LPRINT: LPRINT: RETURN
510 V=VAL (T$): IF V=0 LPRINT" "T$;:RETURN
520 LPRINTUSING "########.##";V;:LPRINT"              ";:RETURN
```

Program Listing 6. Adjustment program for Minicalc.



Searching for answers

Selecting Records

I'd like to select records in a variables file when merging with the master file in Scripsit. Does anyone know of a program that will accomplish this? It would also be nice if the selection criteria could accommodate more than OR or AND connectives.

Paulo Guarinello
Emiliano Perneta, 837/802
Curitiba, Parana
Brazil

Up and Running

Can someone help me get MicroSoft's Level II Basic Compiler up and running on a Model III?

John S. Letcher Jr.
P.O. Box 104
Southwest Harbor, ME 04679

Line-Renumbering Program

I have a Model III cassette operating system. After writing lengthy programs in Basic, I desperately need a line renumbering program. Does anyone know where I can get such a program?

Robert Leland
737 Knollwood Lane
San Dimas, CA 91773

Sell Me Yours

I can't find Microsoft's Extended muMath for the Model III anywhere. Does anyone know where I can get a copy? If you're willing to sell yours, contact me.

Kenneth Meyer
1314 Ault View Ave.
Cincinnati, OH 45208

Needs a Conversion Program

I have been using a Wang 2200 B system for several years, and now that I have a Model II, I have no way to access my engineering programs on the Wang disks. I'd like to know if there's a reformatter/conversion program available that can exchange data files from Wang to TRSDOS and vice versa.

Julio E. Sosa
P.O. Box 6-473
Panama City
Republic of Panama

A Call from Nature

I'm a teacher at a natural resource college, and I'd like to hear from others on how they use their TRS-80s in the natural resource fields. Also, I'd like to know how to get double-width characters in Assembly language.

Robert Johnson
274 Grey St.
Brantford, Ontario N3S 4W8
Canada

Footnotes[1]

I'm looking for a Model I footnote program integrated with Scripsit that features pagination, formatting, and numbering. Can anyone help me?

Also, has anyone figured out how to save either a Basic or ASCII file onto tape from the Atari 400 and then read that tape into a Model I?

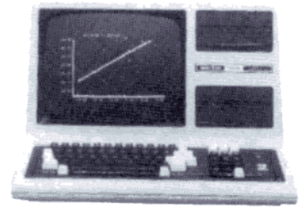
E. Judson Jennings
299 Ridgewood Ave.
Glen Ridge, NJ 07028

From Computer Plus to YOU . . .

PLUS after PLUS after PLUS

Model 100 8K $679
Model 100 24K $835

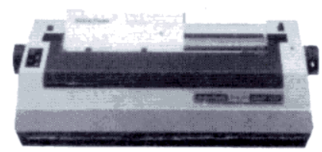
Color Computer II 16K $185
w/16K Ext. Basic $245
w/64K Ext. Basic $305



Model IV 16K $849
Model IV 64K
2 Disk & RS232 $1699

CITOH Prowriter $375
CITOH Prowriter II $649

CoCo Drive 0 $329
CoCo Drive 1 $235



DWP210 $629

BUY DIRECT

Here are just a few of our fine offers . . . call TOLL FREE for full information.

COMPUTERS

Model 12 64K 1 Drive	$2699
Model 12 64K 2 Drive	3375
Model 4 16K	849
Model 4 64K	
2 Disk & RS232	1699
Model 4 Portable	
64K 2 Disk	1525
Color Computer II 16K	145
w/16K ext. basic	210
w/64K ext. basic	305
Pocket Computer 2	165
Model 16B 1Dr 256K	4249
Model 16B 2Dr 256K	4915
Model 100 8K	679
Model 100 24K	835

MODEMS

Hayes Smartmodem II	235
Hayes Smartmodem 1200	565
Novation Smartcat 1200	459
Novation J-Cat	125
R.S. AC-3	129
R.S. Modem I	89
R.S. Modem II	160

PRINTERS

Smith Corona TPI D.W.	469
Silver Reed EXP500 D.W.	425
Silver Reed EXP550 D.W.	645
Daisy Wheel II	1745
DWP210	629
DWP410	1159
CGP115	159
CGP220 Ink Jet	545
DMP100	215
DMP120	395
DMP200	520
DMP420	735
DMP500	1219
DMP2100	1745
Gemini 10X	315
Delta-10	515
Gemini 15	425
CITOH Prowriter	375
CITOH Prowriter II	649
Okidata	CALL
Epson	CALL

DISK DRIVES

R.S. Model 4 Drive 0	515
Color Computer Drive 0	329
Color Computer Drive 1	235
Primary Hard Disk M12	2689
Primary Hard Disk MIII	1799

ETC.

CCR-81 recorder	52
C.C. Joysticks (pair)	22
16K Ram Chips	25
64K Ram Chips	75
8K Par/Par Microfaser	135
Printer Cables	
Printer Stands	
Printer Ribbons	
Printer Paper	
Dust Covers	
Computer Books	
R.S. Software 10% off list.	
Send for listing of brand name software.	

We have the lowest possible Fully Warranteed Prices AND a full complement of Radio Shack Software.

TOLL FREE 1-800-343-8124

computer plus

P.O. Box 1094
480 King Street
Littleton, MA 01460
617-486-3193

Write for your free catalog

✓ 18

SINCE 1973 —

Prices subject to change without notice.
Not responsible for typographical errors.
TRS-80 is a registered trademark of Tandy Corp.

✓ See List of Advertisers on Page 291

Flaws and fixes

Now You Can Talk

There is an error in Program Listing 5 of Douglas Payne's article, "Talk to the Big Guys" (September 1983, p. 230). In the program UT5, lines 1490, 1510, 1580, 1630, 1650, 1700, 1720 have been incorrectly printed as DEFB '''. You must change them to read DEFB '27H' for the program to work correctly.—*Eds.*

Blackjack

The last part of line 1230 and all of line 1240 is missing in Program Listing 4 of my "Model II Casino" article (August 1983, p. 148). These lines should read:

```
1230 PRINTCHR$(2);:PRINT@L," "
:PRINT@L+80," ":PRINT@L+160," "
:PRINTCHR$(1);:RETURN
1240 PRINTCHR$(2);:L=683:GOSUB1230
:RX=R6:GOSUB1160:T$=M$(R6)
:GOSUB1200:PRINTCHR$(1);:RETURN
```

Also, in the "Black Friday" conversion program (Take II, July 1983, p. 342), the last equals sign in line 510 should be an asterisk.

Byron Lott
913 Inverness Way
Sunnyvale, CA 94087

A Random Breakout?

There is a minor problem with J.B. Harrell's "Fortran Breakout" article (July 1983, p. 186). The IRAND function in USRLIB/MAC isn't very random after the first two or three calls (to be sure, I checked the routine with TASMON). It does have an easy fix, however. Just call RANDOM every time (see Fig. 1).

In addition, three bytes can be saved for each of the one-command calls (CLS, RANDOM, GETCH) by simply setting the entry equal to the address (ex. CLS EQU 01C9H).

Mary Jo Kostya
Oppenheimer Str. 17
D-6000 Frankfurt 70
Federal Republic of Germany

```
IRAND:  PUSH  HL
        CALL  RANDOM   ;reinitialize random numbers
        POP   HL
        LD    A,(HL)   ;and so on
```

Figure 1. Calling Random.

The Answer is...
NEWSCRIPT!™

THE WORD PROCESSOR FOR BUSINESSMEN AND PROFESSIONALS

With ongoing support directly from us

A FEW OF NEWSCRIPT's 200 STANDARD FEATURES:

- FORM LETTERS WITH MERGING OF NAMES AND ADDRESSES
- GIVES SUPERB APPEARANCE TO YOUR FINAL DOCUMENTS
- COMPREHENSIVE MANUAL WITH TUTORIAL AND EXAMPLES
- CENTERING, TOP/BOTTOM TITLES, INDENTS, PAGINATION
- UNDERLINING, BOLDFACE, DOUBLE-WIDTH, ITALICS†
- SUB/SUPER SCRIPTS, RIGHT-JUSTIFIED PROPORTIONAL†
- CREATES TABLE OF CONTENTS, SORTED INDEX
- "LEGAL" LINE NUMBERING
- SCREEN GRAPHICS, SPECIAL PRINTER SYMBOLS†
- SEARCH/REPLACE GLOBALLY OR WITHIN LINES, COLUMNS
- BLOCK MOVE, COPY, DELETE, INSERT, FILE MERGES
- AUTOSAVE, WHOOPS, DIRECTORY, KILL
- SUPPORT FOR **ALL** LISTED PRINTERS IS INCLUDED ** (**NO PATCHES INVOLVED**) **
- SUPPLIED READY-TO-RUN ON "TINY" DOSPLUS
- ALSO RUNS UNDER NEWDOS/80, LDOS, MULTIDOS, TRSDOS

NEWSCRIPT 7.1:	$124.95
Mailing Labels Option:	29.95
Special: NEWSCRIPT + LABELS:	139.95
Daisywheel Proportional Option:	49.95
"Pencil"/"Scripsit" File Convertor:	24.95
NEWSCRIPT Manual & Reference card only:	29.95
Electric Webster + Correction Feature:	149.50
Hyphenation Feature for Electric Webster:	49.95
Grammatical Feature for Electric Webster:	39.95
Dotwriter 3.0:	79.95
Dotwriter + Letter Utilities:	99.95
6.0 PLUS-MSS Model 4 utility pak	49.95

REQUIRED CONFIGURATION:
48K TRS-80, MAX-80, LNW, or compatible, with one or more disk drives. Specify Model I or Model III.

† some features work only if your printer has the mechanical capability.

** Daisy Wheel Proportional is an extra-cost option.

TO ORDER, CALL NOW, TOLL-FREE: (800) 824-7888, Operator 422

For orders, information, or names of nearby dealers: (213) 764-3131, or write to us.

Order from your Software dealer or from:

PROSOFT®

98

Dep't. C, Box 560 No. Hollywood, CA 91603

TERMS: VISA, Mastercard, checks, money orders, COD. No P.O.'s accepted. Most orders shipped within 24 hours. Please add $3.00 for surface UPS in U.S.A., or $6.00 for UPS Blue Label. Add $6.00 in Canada, $15.00 overseas air shipment, 6½% sales tax in California.

BUILT-IN SUPPORT FOR MOST POPULAR PRINTERS†, INCLUDING:

Anadex, Brother, Centronics, C.Itoh, Diablo, Epson, Gemini, Microline, NEC, Prowriter, Qume, Radio Shack (LP 1-8, DW2, DMP-410, DWP 200-2100), Smith Corona, Teletype, Typewriter, anything compatible with any of these, and many others, parallel and RS-232.

SPECIAL AVAILABLE OPTION: Right-justified proportional for Diablo, F-10, Qume, Spinwriter, etc. Requires "Daisywheel Proportional" Option plus NEWSCRIPT.

REVIEWERS AND USERS AGREE:

"NEWSCRIPT" is the best word procesor I have seen . . . unsurpassed in printer control . . . no other TRS-80 word processor can match its ability to format text . . . its editor is fast, easy, and powerful." *(80 MICRO, Oct. 1982)*

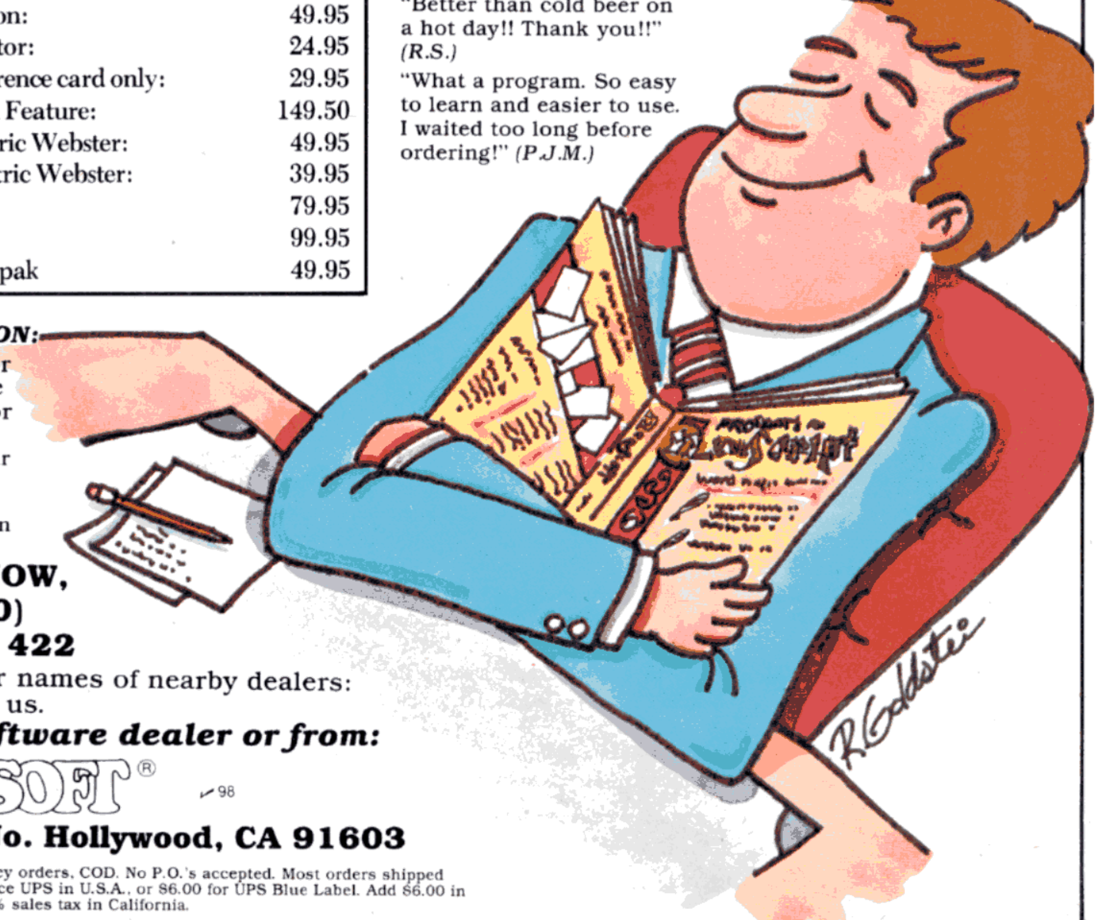
"Your phone information system and the prompt and courteous staff that you provide to help your clients . . . are worth the cost of the system." *(V.H.H.)*

"Better than cold beer on a hot day!! Thank you!!" *(R.S.)*

"What a program. So easy to learn and easier to use. I waited too long before ordering!" *(P.J.M.)*

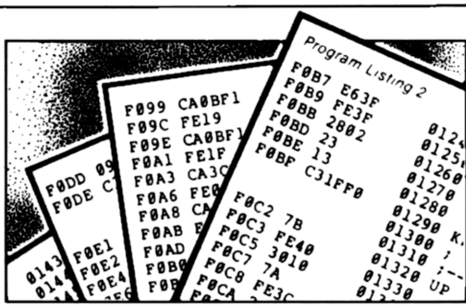
". . . takes the TRS-80 to a new level of text handling . . . very user-friendly . . . superb documentation, adaptability to many printers and operating systems . . . a standard against which other TRS-80 word processing programs will be judged." *(SOFTSIDE, Dec. 1982)*

". . . ongoing support second to none, with superb documentation." *(80 U.S. Journal, Feb. 1982)*



THE NEXT STEP

by Hardin Brothers



Displaying the values of variables

Last month, I explained a simple utility that displayed all current variables in a running Basic program. This month, as promised, I'll explain how to add a new feature to that utility: the ability to display the current value of any current variable.

After the program displays each screen of active variables, you'll be able to either press the enter key to go on to the next page or type in a variable name. If you enter a variable name, its value appears on the screen until you press a key; then you again face the choice of pressing the enter key or typing in another variable name.

Much of Program Listing 1 is identical to last month's program. Realizing you probably won't want to type in the same program again, I've marked all new or modified lines with an asterisk at the beginning of the comment column. Some line numbers have changed, but if a program block doesn't have any marked lines, you won't need to change anything in the source code. You will, however, have to delete lines 2550 and 2610–2670 from last month's program. After you've added the new lines, type N100,10 and be sure you end up with the same last line as the listing.

The first major change in the program, aside from some new EQU statements and a new ORG address, occurs in lines 960–1040, where I've added a flag-handling routine. This program uses ROM routines to read and decipher the keyboard. However, the entire program starts with decoding a special combination of three keys: the shift, down-arrow, and V keys. If the utility is active (displaying variables) and the operator again pushes the control/V key combination, the program doubles back on itself recursively. The screen and cursor buffers are overwritten, and the program ends up in an endless loop.

To avoid such problems, line 970 reads a flag that shows whether the utility program is currently running. If the program sets the 1-byte flag to zero, it ignores control/V and invokes the regular keyboard driver. If the utility program sets the flag to any other value (generally 0FF hexadecimal (hex)) it knows that it's not currently running and lets itself be invoked. The utility sets the flag to zero when you enter the program and resets to 0FF hex during the Done routine as the program returns control to Basic.

The flag avoids the problems that recursion would create, but it can cause another problem: Some ROM routines that the program calls generate an error condition. If an error occurs, the ROM routine reports the error on the screen and returns control to Basic's READY prompt, bypassing the normal flag reset in Done. The flag remains set at zero and you won't be able to use the utility because it reacts as if it's already being used. To reactivate the program, you have to POKE any non-zero value into the flag's location (0BFFF hex if you don't change the ORG value).

The second addition to the program is the routine called NOKEY. I found that I occasionally fail to release the control/V key sequence quickly enough, so one of the keys is read as part of the variable name. The NOKEY routine guards against this by refusing to let the program start until you've released all the keys. The B register is used as a counter to help delay the NOKEY reading to avoid keybounce. If you still have a problem with spurious key reads, increase the time of the loop by using BC as a counter.

The ENDSCR routine is almost entirely new and contains the crux of this

Address	**Description**
RST 18 hex	Compares the values in the DE and HL register pairs. Sets flags to show results: Carry HL<DE No Carry HL>=DE Zero HL=DE Not Zero HL<>DE
0049 hex	Waits for single keyboard input—returns ASCII value of input key in A register. Uses AF and DE registers.
01C9 hex	Clears screen, resets cursor to top of screen, sets 64-character mode, and turns off cassette port (Basic's CLS routine).
033A hex	Displays character in A on screen at next print position. Uses AF.
05D9 hex	Inputs line from keyboard. On entry, HL points to buffer to hold input, B holds maximum number of characters to input. You terminate input by pressing the enter or break key. On output, HL points to start of text, B equals actual input length (except terminator), C holds original value of B, A holds ASCII value of terminating character, C flag is set only if input is terminated with the break key.
0A9A hex	Current value in HL is loaded into Basic's workspace/accumulator, and the variable type flag is set to 02 (integer).
0ACC hex	Value in workspace/accumulator changed from integer to single precision.
0FDB hex	Converts value in work workspace/accumulator into display form, and returns address of resulting string in HL.
2540 hex	Loads value of variable into workspace/accumulator. On entry, HL points to first character of variable name. On exit, HL points to first character following variable name.
2B75 hex	Output a string to current output device. On entry, HL points to first character of string, which must end with a zero byte.

Figure. ROM routines used in the variable display programs.

month's modification. After correctly positioning the print cursor, the program prints a CHR$(1F hex) on the screen. That control character clears the screen from the present print position to the end of the screen. This is necessary to erase any characters left from a previous variable and value display.

Then, after the program displays the prompt message, line 2830 points HL at an input buffer and sets B to the buffer's length in preparation for accepting a typed-in variable name. Since both DOS and Basic have input buffers, you may wonder why they weren't used instead of dedicating space to a third buffer. I didn't use the DOS buffer because there is no guarantee it will be in the same position with different DOS systems, or with different versions of the same DOS. I left the Basic buffer alone because you might want to call up the variable display while writing or editing a line of Basic and not wish to return to a line filled with garbage.

After you set up HL and B, the program calls the ROM routine LINEIN. This routine accepts keyboard input of a string up to the maximum length set in register B. With each keystroke, the character appears on the screen and is stored in the buffer to which HL points. The routine reacts normally to backspacing and other control characters. On return, the B register contains the number of characters actually entered except for the terminating character (enter or break). If you press the break key, the program sets the carry flag.

If you press the break key during variable input and set the carry flag, the program loops back to the prompt and again awaits operator input. If B equals zero, you pressed the enter key without any additional characters, so the screen clears and control returns to the display routines.

If you input any additional characters, the program assumes it has a variable name in the buffer. First, the program establishes a new cursor position. Then it prints the variable name followed by an equal sign. The program points HL at the variable name again and calls the ROM routine VALACU.

VALACU (2540 hex) is a complex routine that takes the variable name at which HL points, evaluates that variable, transfers its value to the low memory accumulator, and sets a flag to indicate the variable type (integer, string,

Program Listing 1. Current variable display.

```
               00100 ;********************************
               00110 ;*                              *
               00120 ;*   Current Variable Display   *
               00130 ;*                              *
               00140 ;*   On <Shift> <Dwn-arrow> <V> *
               00150 ;*   shows two screens (or more)*
               00160 ;*   1st -- all defined, simple *
               00170 ;*         variables            *
               00180 ;*   2nd -- all defined array   *
               00190 ;*         variables            *
               00200 ;*                              *
               00210 ;*   After each screen, values  *
               00220 ;*   of current variables may   *
               00230 ;*   be listed.                 *
               00240 ;*                              *
               00250 ;********************************
               00260 ;
               00270 ;
               00280 ;Listing of routines and addresses:
               00290 ;
033A           00300 PRINT1   EQU     033AH           ;PRINT 1 CHAR.
2B75           00310 PRINT    EQU     2B75H           ;PRINT STRING
01C9           00320 CLS      EQU     01C9H           ;CLEAR SCREEN
0A9A           00330 HLACUM   EQU     0A9AH           ;HL==> RAM ACCUMULATOR
0ACC           00340 ACINSN   EQU     0ACCH           ;ACCUM. VALUE INT=>SINGLE
0FBD           00350 ACUSTR   EQU     0FBDH           ;ACCUM => ASCII STRING
05D9           00360 LINEIN   EQU     05D9H           ;*ROM LINE INPUT ROUTINE
2540           00370 VALACU   EQU     2540H           ;*VALUE OF VAR.=> ACCUM
40AF           00380 VTFLAG   EQU     40AFH           ;*VARIABLE TYPE MARKER
0049           00390 WAITKY   EQU     0049H           ;*GET SINGLE KB INPUT
               00400 ;
4016           00410 KBDVR    EQU     4016H           ;KB DRIVER ADDRESS
4020           00420 CURSAD   EQU     4020H           ;CURSOR POSITION ADDRESS
               00430 ;
3804           00440 ROW3     EQU     3804H           ;P-W KEYBOARD ROW
3840           00450 ROW7     EQU     3840H           ;ENT & ARROWS KB ROW
3880           00460 ROW8     EQU     3880H           ;SHIFT KEYBOARD ROW
               00470 ;
3C00           00480 VIDEO    EQU     3C00H           ;TOP OF SCREEN
3F7B           00490 ENDDSP   EQU     3F7BH           ;LAST LINE FOR DISPLAY
3F88           00500 PRTPOS   EQU     3F88H           ;ADDR. FOR PROMPT DISPLAY
               00510 ;
4411           00520 MEMTOP   EQU     4411H           ;MEMTOP FOR MOD.III DISK
               00530 ;        use 4049H for MODI Disk, 40B1H for tape systems
               00540 ;
40F9           00550 SVT      EQU     40F9H           ;BEG. OF VARIABLE LIST
40FB           00560 AVT      EQU     40FBH           ;BEG. OF ARRAY LIST
40FD           00570 FREE     EQU     40FDH           ;BEG. OF FREE SPACE
               00580 ;
               00590 ;
               00600 ; First, patch routine into keyboard
               00610 ;  driver, being careful to save current
               00620 ;  driver address, and reset MEMTOP.
               00630 ;
B9C5           00640          ORG     0B9C5H          ;CHANGE ORG TO RELOCATE
               00650 ;                this address for top of 32K RAM
               00660 ;
B9C5 2A1640    00670 SETUP    LD      HL,(KBDVR)      ;GET CURRENT DRIVER ADDR.
B9C8 11DDB9    00680          LD      DE,TEST         ;GET ROUTINE ADDR.
B9CB DF        00690          RST     18H             ;COMPARE DE & HL
B9CC 280C      00700          JR      Z,SET10         ;GO IF ALREADY SET
B9CE 22F0B9    00710          LD      (RETURN+1),HL   ;SET RETURN ADDR.
B9D1 ED531640  00720          LD      (KBDVR),DE      ;SET 'TEST' AS KB DRIVER
B9D5 1B        00730          DEC     DE              ;DE=TEST-1
B9D6 ED531144  00740          LD      (MEMTOP),DE     ;SET MEMTOP
B9DA C32D40    00750 SET10    JP      402DH           ;RETURN TO DOS
               00760 ;
               00770 ;  Line 750 should be  JP 1A19H for tape systems
               00780 ;
               00790 ;  Now test for <SHIFT> <DOWN-ARROW> <V>
               00800 ;
B9DD 3A4038    00810 TEST     LD      A,(ROW7)        ;GET DWN-ARW ROW
B9E0 E610      00820          AND     10H             ;IS IT DWN-ARW?
B9E2 280B      00830          JR      Z,RETURN        ;GO IF NOT
B9E4 3A8038    00840          LD      A,(ROW8)        ;GET SHIFT ROW
B9E7 B7        00850          OR      A               ;SET FLAGS
B9E8 2805      00860          JR      Z,RETURN        ;GO IF NO SHIFT KEY
B9EA 3A0438    00870          LD      A,(ROW3)        ;GET P-W ROW
B9ED E640      00880          AND     40H             ;CHECK FOR V
B9EF CA0000    00890 RETURN   JP      Z,0000H         ;SETUP PUTS VALUE HERE
               00900 ;
               00910 ; <Shift> <down-arrow> <V> have been pressed.
               00920 ; Check flag status.
               00930 ; Save all registers.
               00940 ; Save screen & then clear it.
               00950 ;
B9F2 CD25BB    00960          CALL    PUSHAL          ;SAVE ALL REGISTERS
B9F5 3AFFBF    00970          LD      A,(FLAG)        ;*GET FLAG REGISTER
B9F8 B7        00980          OR      A               ;*SET F REGISTER
B9F9 2806      00990          JR      Z,NOTNOW        ;*GO IF ZERO
B9FB AF        01000          XOR     A               ;*SET A TO ZERO
B9FC 32FFBF    01010          LD      (FLAG),A        ;*SET FLAG REGISTER
B9FF 1805      01020          JR      NOKEY           ;*AND GO
               01030 ;
BA01 CD2EBB    01040 NOTNOW   CALL    POPAL           ;*RESTORE REGISTERS
BA04 18E9      01050          JR      RETURN          ;*GO TO NORMAL KB DRIVER
               01060 ;
BA06 06FF      01070 NOKEY    LD      B,255           ;*SET FOR TIMING LOOP
```

Listing 1 continued

Listing 1 continued

```
BA08 3AFF3B   01080 NOKEY5  LD      A,(3BFFH)       ;*CHECK FOR ANY KEY
BA0B B7       01090         OR      A               ;*SET FLAGS
BA0C 20FA     01100         JR      NZ,NOKEY5       ;*LOOP UNTIL ALL KEYS UP
BA0E 10F8     01110         DJNZ    NOKEY5          ;*LOOP FOR KEY BOUNCE
              01120 ;
BA10 11FDBB   01130 SAVE    LD      DE,SCRBUF       ;DE==> SCREEN BUFFER
BA13 21003C   01140         LD      HL,VIDEO        ;HL==> SCREEN
BA16 010004   01150         LD      BC,400H         ;BC = SCREEN LENGTH
BA19 EDB0     01160         LDIR                    ;MOVE SCREEN
BA1B 2A2040   01170         LD      HL,(CURSAD)     ;GET CURRENT CURSOR ADDR.
BA1E 22FDBF   01180         LD      (CRSBUF),HL     ;AND SAVE IT ALSO
BA21 CDC901   01190         CALL    CLS             ;CLEAR THE SCREEN
BA24 21883F   01200         LD      HL,PRTPOS       ;GET PRINT POSITION
BA27 222040   01210         LD      (CURSAD),HL     ;MOVE CURSOR AWAY
              01220 ;
              01230 ; Now get list of simple (not array) variables and print
              01240 ;  each on the screen.
              01250 ;
BA2A DD2AF940 01260         LD      IX,(SVT)        ;IX==>VARIABLE LIST
BA2E 21003C   01270         LD      HL,VIDEO        ;HL==> SCREEN
BA31 FD21E2BB 01280         LD      IY,TYPES        ;IY==> VAR. TYPE TABLE
BA35 E5       01290 VAR10   PUSH    HL              ;SAVE SCREEN PTR.
BA36 DDE5     01300         PUSH    IX              ;MOVE VAR. LIST PTR
BA38 E1       01310         POP     HL              ;  TO HL
BA39 ED5BFB40 01320         LD      DE,(AVT)        ;GET END OF VAR. TABLE
BA3D DF       01330         RST     18H             ;CP HL:DE
BA3E E1       01340         POP     HL              ;RECOVER SCREEN PTR.
BA3F 303E     01350         JR      NC,ARRAY        ;GO IF NO MORE VARIABLES
              01360 ;
BA41 DD4600   01370         LD      B,(IX)          ;GET VARIABLE TYPE
BA44 DD7E02   01380         LD      A,(IX+2)        ;1ST LETTER OF VAR. NAME
BA47 77       01390         LD      (HL),A          ;PRINT IT
BA48 23       01400         INC     HL              ;BUMP POINTER
BA49 DD7E01   01410         LD      A,(IX+1)        ;2ND LETTER OF VAR. NAME
BA4C B7       01420         OR      A               ;SET FLAGS
BA4D 2802     01430         JR      Z,VAR20         ;GO IF ZERO
BA4F 77       01440         LD      (HL),A          ;ELSE PRINT IT
BA50 23       01450         INC     HL              ;AND BUMP POINTER
BA51 78       01460 VAR20   LD      A,B             ;GET VARIABLE TYPE
BA52 3D       01470         DEC     A               ;DECREASE BY TWO TO
BA53 3D       01480         DEC     A               ;  ALIGN WITH TABLE
BA54 3259BA   01490         LD      (IYPTR+2),A     ;USE FOR OFFSET
BA57 FD7E00   01500 IYPTR   LD      A,(IY+0)        ;GET TYPE SYMBOL
BA5A 77       01510         LD      (HL),A          ;PUT ON SCREEN
BA5B 23       01520         INC     HL              ;BUMP POINTER
BA5C 23       01530         INC     HL              ;AND AGAIN FOR SPACE
              01540 ;
BA5D DD23     01550         INC     IX              ;GET PAST HEADER
BA5F DD23     01560         INC     IX              ;  WITH THREE
BA61 DD23     01570         INC     IX              ;  INCREMENTS
BA63 DD23     01580 VAR30   INC     IX              ;MOVE PAST VAR. INFO
BA65 10FC     01590         DJNZ    VAR30           ; DEPENDING ON VAR. TYPE
              01600                                 ;IX==> NEXT VARIABLE
              01610 ;Check screen
BA67 117B3F   01620         LD      DE,ENDDSP       ;LAST PRINT POS.
BA6A DF       01630         RST     18H             ;CP HL:DE
BA6B D437BB   01640         CALL    NC,ENDSCR       ;GO IF SCREEN FULL
BA6E 7D       01650         LD      A,L             ;GET LSB OF SCREEN PTR.
BA6F E63F     01660         AND     3FH             ;MASK BITS 6 & 7
BA71 FE3C     01670         CP      3CH             ;END OF LINE?
BA73 38C0     01680         JR      C,VAR10         ;RETURN IF NOT
BA75 114000   01690         LD      DE,40H          ;LINE OFFSET
BA78 19       01700         ADD     HL,DE           ;HL==> NEXT LINE
BA79 7D       01710         LD      A,L             ;GET LSB
BA7A E6C0     01720         AND     0C0H            ;MASK OUT BITS 0-5
BA7C 6F       01730         LD      L,A             ;HL==> START OF NEXT LINE
BA7D 18B6     01740         JR      VAR10           ;GET ANOTHER VARIABLE
              01750 ;
              01760 ;Now show arrays
              01770 ;
BA7F CD37BB   01780 ARRAY   CALL    ENDSCR          ;NEW SCREEN FOR ARRAYS
BA82 DD2AFB40 01790         LD      IX,(AVT)        ;IX==>ARRAY TABLE
BA86 DDE5     01800 ARR10   PUSH    IX              ;TRANSFER PTR TO
BA88 E1       01810         POP     HL              ;  TO HL
BA89 ED5BFD40 01820         LD      DE,(FREE)       ;DE==>END OF ARRAYS
BA8D DF       01830         RST     18H             ;CP HL:DE
BA8E 3075     01840         JR      NC,DONE         ;GO IF END OF TABLE
              01850 ;
BA90 DD4600   01860         LD      B,(IX+0)        ;GET VARIABLE TYPE
BA93 DD7E02   01870         LD      A,(IX+2)        ;1ST LETTER OF VAR. NAME
BA96 CD3A03   01880         CALL    PRINT1          ;PRINT A
BA99 DD7E01   01890         LD      A,(IX+1)        ;2ND LETTER OF VAR. NAME
BA9C B7       01900         OR      A               ;SET FLAGS
BA9D 2803     01910         JR      Z,ARR20         ;GO IF 0
BA9F CD3A03   01920         CALL    PRINT1          ;ELSE PRINT IT
BAA2 78       01930 ARR20   LD      A,B             ;GET VAR. TYPE VALUE
BAA3 3D       01940         DEC     A               ;SUBTRACT TWO TO
BAA4 3D       01950         DEC     A               ;  ALIGN WITH TABLE
BAA5 32AABA   01960         LD      (IYPTR2+2),A    ;ADDRESS TABLE
BAA8 FD7E00   01970 IYPTR2  LD      A,(IY+0)        ;GET TYPE SYMBOL
BAAB CD3A03   01980         CALL    PRINT1          ;AND PRINT IT
BAAE 3E28     01990         LD      A,'('           ;PAREN. CHAR.
BAB0 CD3A03   02000         CALL    PRINT1          ;AND PRINT IT
              02010 ;
BAB3 DD4E05   02020         LD      C,(IX+5)        ;GET # OF DIMENSIONS
BAB6 DD5E03   02030         LD      E,(IX+3)        ;DE WILL HAVE OFFSET
BAB9 DD5604   02040         LD      D,(IX+4)        ;  TO NEXT ARRAY
BABC DDE5     02050         PUSH    IX              ;TRANSFER IX VALUE TO
BABE E1       02060         POP     HL              ;  HL REGISTER
BABF 19       02070         ADD     HL,DE           ;ADD OFFSET
```

Listing 1 continued

single precision, or double precision). If the variable is a string, its VARPTR address is loaded into the accumulator instead of its value.

Next, the program checks the variable type flag. If the type flag equals 3, the variable is a string, and control passes to STRDSP in line 3140. Otherwise, the ROM routine ACUSTR, which I described last month, changes the variable's value into an ASCII string and the program prints that string. Then the ROM routine WAITKY (0049 hex), which waits for any keyboard input, is used to freeze the display until you press any key before the program loops back to END10. WAITKY takes the place of last month's GETENT, which is no longer included in the program.

If the variable is a string, IX points to its VARPTR address (line 3210). Then the program loads B with the string's length and HL with its location in memory. Because the string is not terminated with a 00 hex byte, you cannot use the normal Print routine. Instead, the program calls the PRTSTR routine at line 3280 to print the string.

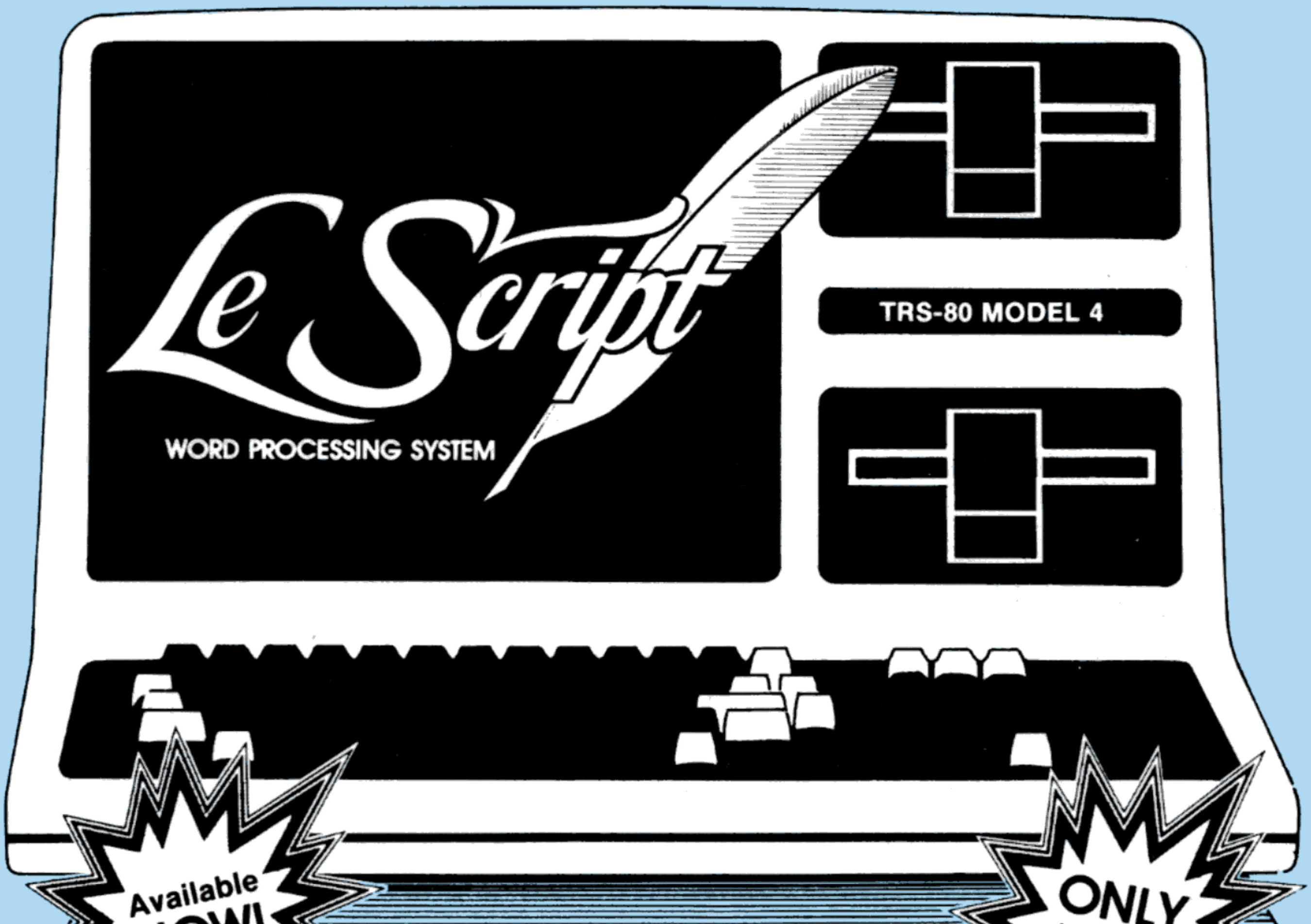
PRTSTR first checks the string's length. If it is zero, control returns immediately without any further processing. Otherwise, the program prints the string with a DJNZ loop until B is decremented to zero.

For quick reference, the Figure summarizes the ROM routines and low-memory pointers used by this month's and last month's programs. Before you add the ROM routines to your own programs, you must exercise caution. Some ROM routines (2B75 hex is a notorious example) only work properly if Basic is initialized. If you use a tape system, you never have to worry because Basic is initialized automatically when you turn on the power.

However, if you use a disk system, you can use programs with ROM routines only if you've loaded Disk Basic since you turned on the computer. If you find a program, particularly a magazine program, that doesn't work as it should, try going to Disk Basic, returning to DOS, and rerunning it.

The problem is that ROM Basic was written to be expanded in two ways. Each expansion necessitates an exit from ROM to a patch point in low memory. The first type of expansion is the addition of Disk Basic command verbs (CMD", INSTR, OPEN, DEF,

TRS-80 MODEL 4
Word Processing Program



Available NOW!

ONLY $129[95]

80 x 24 DISPLAY

- Works with **TRSDOS 6.0**
- **90K TEXT BUFFER** when expanded memory is installed.
- Built-in **FORM LETTER** and **MAIL-MERGE** capabilities.
- Performs **PROPORTIONAL-SPACE** right-margin justification on over 60 different printers (all drivers included **FREE**).
- Same powerful editing features as **ZORLOF II,** with dozens more features added.
- Versions also available to run on **MODEL I, MODEL III, LNW-80, PMC-80,** and **MAX-80** (80x24 display).

FREE

- 30-Day Support to Registered Owners
- Subscription to LeScript Newsletter

LeScript costs $200 less than competing products matching its capabilities.

VISA MasterCard

CALL 1-305-259-9397

ANITEK

ANITEK SOFTWARE PRODUCTS
P.O. BOX 361136
MELBOURNE, FL 32936

✓141

FREE SHIPPING WITHIN THE U.S.; OUTSIDE THE U.S. ADD $10.00 FOR SHIPPING; FLORIDA RESIDENTS ADD 5% SALES TAX. ALL ORDERS PREPAID BY CHECK, MONEY ORDER, CREDIT CARD OR C.O.D.

Listing 1 continued

```
BACØ 11Ø5ØØ                LD     DE,5           ;OFFSET FOR HEADER
BAC3 19                    ADD    HL,DE          ;HL==> NEXT ARRAY
BAC4 E5                    PUSH   HL             ;SAVE ADDRESS
                   ;
BAC5 Ø6Ø6                  LD     B,6            ;BUMP IX 6 TIMES
BAC7 DD23          ARR3Ø   INC    IX             ;  SO IX==> SIZE OF
BAC9 1ØFC                  DJNZ   ARR3Ø          ;  1ST DIMENSION
                   ;
BACB 41                    LD     B,C            ;GET # OF DIM.
BACC CB2Ø                  SLA    B              ;MULTIPLY BY TWO
BACE DD23          ARR4Ø   INC    IX             ;BUMP POINTER
BADØ 1ØFC                  DJNZ   ARR4Ø          ;UNTIL PAST DIM SIZES
                   ;
BAD2 41                    LD     B,C            ;GET # OF DIM. AGAIN
BAD3 DD2B          ARR5Ø   DEC    IX             ;DROP IX UNTIL IT
BAD5 DD2B                  DEC    IX             ;  POINTS TO NEXT DIM.
BAD7 DD6EØØ                LD     L,(IX+Ø)       ;GET LSB OF DIM SIZE
BADA DD66Ø1                LD     H,(IX+1)       ;  AND MSB
BADD 2B                    DEC    HL             ;CORRECT FOR Ø ELEMENT
BADE CDB2BB                CALL   ASCPRT         ;PRINT AS ASCII
                   ;
BAE1 3E2C                  LD     A,','          ;GET COMMA CHAR.
BAE3 CD3AØ3                CALL   PRINT1         ;AND PRINT IT
BAE6 1ØEB                  DJNZ   ARR5Ø          ;REPEAT FOR ALL DIM.S
                   ;
BAE8 21204Ø                LD     HL,CURSAD      ;HL==>CURSOR POS'N
BAEB 35                    DEC    (HL)           ;MOVE BACK OVER LAST ","
BAEC 3E29                  LD     A,')'          ;GET CLOSE PAREN.
BAEE CD3AØ3                CALL   PRINT1         ;AND PRINT IT
BAF1 3E2Ø                  LD     A,' '          ;NOW AN ASCII SPACE
BAF3 CD3AØ3                CALL   PRINT1         ;AND PRINT IT
BAF6 2A2Ø4Ø                LD     HL,(CURSAD)    ;GET CURSOR POS'N
BAF9 117B3F                LD     DE,ENDDSP      ;END OF PRINT AREA
BAFC DF                    RST    18H            ;CP HL:DE
BAFD D437BB                CALL   NC,ENDSCR      ;GO IF SCREEN FULL
                   ;
BBØØ DDE1                  POP    IX             ;IX==> NEXT ARRAY
BBØ2 C386BA                JP     ARR1Ø          ;GO TO WORK ON IT
                   ;
BBØ5 CD37BB        DONE    CALL   ENDSCR         ;ASK FOR ENTER
BBØ8 21FDBB                LD     HL,SCRBUF      ;HL==>SCREEN BUFFER
BBØB 11ØØ3C                LD     DE,VIDEO       ;DE==>SCREEN
BBØE Ø1ØØØ4                LD     BC,4ØØH        ;BC = SCREEN LENGTH
BB11 EDBØ                  LDIR                  ;MOVE TO SCREEN
BB13 2AFDBF                LD     HL,(CRSBUF)    ;GET OLD CURS. POS'N
BB16 22204Ø                LD     (CURSAD),HL    ;AND RESTORE IT
BB19 3EFF                  LD     A,ØFFH         ;*VALUE FOR RECUR. FLAG
BB1B 32FFBF                LD     (FLAG),A       ;*RESET RECURS. FLAG
BB1E CD2EBB                CALL   POPAL          ;RESTORE REGISTERS
BB21 AF                    XOR    A              ;A & Z-FLAG SHOW Ø
BB22 C3EFB9                JP     RETURN         ;RETURN TO BASIC
                   ;
BB25 E3            PUSHAL  EX     (SP),HL        ;HL ON STACK; SAVE RET
BB26 C5                    PUSH   BC             ;SAVE ALL REGS
BB27 D5                    PUSH   DE
BB28 DDE5                  PUSH   IX
BB2A FDE5                  PUSH   IY
BB2C E5                    PUSH   HL             ;ORIG. RET ADDR.TO STACK
BB2D C9                    RET
                   ;
BB2E E1            POPAL   POP    HL             ;GET RET. ADDR.
BB2F FDE1                  POP    IY             ;RESTORE ALL REGS.
BB31 DDE1                  POP    IX
BB33 D1                    POP    DE
BB34 C1                    POP    BC
BB35 E3                    EX     (SP),HL        ;GET ORIG. HL
BB36 C9                    RET
                   ;
BB37 CD25BB        ENDSCR  CALL   PUSHAL         ;SAVE REGISTERS
BB3A 21883F        END1Ø   LD     HL,PRTPOS      ;*HL==>PRINT POSITION
BB3D 22204Ø                LD     (CURSAD),HL    ;SET CURSOR ADDRESS
BB4Ø 3E1F                  LD     A,1FH          ;*1F=CLEAR TO END OF FRAME
BB42 CD3AØ3                CALL   PRINT1         ;*PRINT IT
BB45 21C8BB                LD     HL,MSG         ;HL==>PROMPT MESSAGE
BB48 CD752B                CALL   PRINT          ;PRINT IT
BB4B 21E9BB                LD     HL,VARBUF      ;*HL==>INPUT BUFFER
BB4E Ø614                  LD     B,2ØD          ;*B=MAX. INPUT LENGTH
BB5Ø CDD9Ø5                CALL   LINEIN         ;*GET KEYBOARD INPUT
                   ;
BB53 38E5                  JR     C,END1Ø        ;*GO IF <BREAK> HIT
BB55 78                    LD     A,B            ;*GET INPUT LENGTH
BB56 B7                    OR     A              ;*SET FLAGS
BB57 2ØØA                  JR     NZ,END2Ø       ;*GO IF VARIABLE REQUESTED
BB59 CDC9Ø1                CALL   CLS            ;ELSE CLEAR SCREEN
BB5C CD2EBB                CALL   POPAL          ;RESTORE REGISTERS
BB5F 21ØØ3C                LD     HL,VIDEO       ;HL==>SCREEN TOP
BB62 C9                    RET                   ;AND RETURN
                   ;
BB63 C5            END2Ø   PUSH   BC             ;*SAVE VARIABLE LENGTH
BB64 Ø6Ø8                  LD     B,8            ;*FOR 8 SPACES
BB66 3E2Ø                  LD     A,' '          ;*A=ASCII SPACE
BB68 CD3AØ3        END3Ø   CALL   PRINT1         ;PRINT
BB6B 1ØFB                  DJNZ   END3Ø          ;*  8 SPACES
BB6D C1                    POP    BC             ;*RECOVER LENGTH COUNT
BB6E 21E9BB                LD     HL,VARBUF      ;*HL==>BEGINNING OF VAR.
BB71 E5                    PUSH   HL             ;*AND SAVE IT
BB72 CDA7BB                CALL   PRTSTR         ;*PRINT VAR NAME AS STRING
BB75 3E2Ø                  LD     A,' '          ;*A=ASCII SPACE
BB77 CD3AØ3                CALL   PRINT1         ;*AND PRINT IT
BB7A 3E3D                  LD     A,'='          ;*A= EQUAL SIGN
```

Listing 1 continued

LOAD, and so on). The execution address for each of these new verbs already exists in ROM, but the operation code does not—DOS must supply it. When you boot up a tape system, each Disk Basic command patch point is filled with the L3 error's address.

The second type of patch point to DOS is completely different. Several ROM routines, including many of the output routines, include a call to a low-memory address. When you use a tape system, a return instruction (0C9 hex) is loaded into each of those addresses; when you load Disk Basic, either a return, or a jump to a special Disk Basic routine is loaded into each address. However, when you first boot up DOS, no values are specifically loaded to those patch points and any program that calls them will probably find itself wandering off into oblivion.

The purpose of the second set of patch points is to let DOS writers add new, more powerful features to Basic. But if you are writing Assembly-language programs meant to be compatible with disk systems and run under DOS, and you want to use ROM routines, you should close any patch points your program might come across with a 0C9 hex byte before the ROM routines are called. The patch points are at addresses 41AC hex, 41AF hex, 41B2 hex, and so on up to 41E2 hex, on both the Models I and III.

Defining Variables

Besides showing the current active variables, this month's program can help you improve your Basic programming. Your programs store all simple variables in memory below all array variables. If your program uses arrays, they have to be moved every time you define or use a new simple variable. If one of your first program lines dimensions all arrays, it will slow down your program considerably by the pauses necessary to shift the arrays up in memory whenever you use a new simple variable. Your program will run faster if you define all simple variables before you dimension arrays.

There are two ways to define simple variables at the beginning of the program. The first, and more structured, is to give each a pre-set value (such as A = 0, B = 2, and C = .479). However, if you're going to set many of the variables to zero, or if the program will set

Listing 1 continued

```
BB7C CD3A03              CALL    PRINT1          ;*AND PRINT THAT
BB7F E1                  POP     HL              ;*HL==>VARIABLE AGAIN
BB80 CD4025              CALL    VALACU          ;*VARIABLE'S VALUE=>ACCUM.
                ;
BB83 3AAF40              LD      A,(VTFLAG)      ;*GET VARIABLE TYPE MARKER
BB86 FE03                CP      3               ;*IS IT A STRING?
BB88 280B                JR      Z,STRDSP        ;*GO TO STRING ROUTINE
                ;
BB8A CDBD0F              CALL    ACUSTR          ;*TURN VALUE INTO STRING
BB8D CD752B              CALL    PRINT           ;*AND PRINT IT
BB90 CD4900     END40    CALL    WAITKY          ;*WAIT FOR KEYSTROKE
BB93 18A5                JR      END10           ;*AND REPEAT
                ;
BB95 DD2A2141   STRDSP   LD      IX,(4121H)      ;*IX = STRING'S VARPTR
BB99 DD4600              LD      B,(IX+0)        ;*B=STRING LENGTH
BB9C DD6E01              LD      L,(IX+1)        ;*MSB OF ADDR.
BB9F DD6602              LD      H,(IX+2)        ;*HL==>STRING
BBA2 CDA7BB              CALL    PRTSTR          ;*PRINT THE STRING
BBA5 18E9                JR      END40           ;*AND GO
                ;
BBA7 78         PRTSTR   LD      A,B             ;*GET STRING LENGTH
BBA8 B7                  OR      A               ;*SET FLAGS
BBA9 C8                  RET     Z               ;*RETURN IF NO LENGTH
BBAA 7E         PRT10    LD      A,(HL)          ;*GET CHARACTER
BBAB CD3A03              CALL    PRINT1          ;*AND PRINT IT
BBAE 23                  INC     HL              ;*BUMP POINTER
BBAF 10F9                DJNZ    PRT10           ;*LOOP UNTIL DONE
BBB1 C9                  RET                     ;*RETURN TO CALLER
                ;
                ;
BBB2            ASCPRT   EQU     $               ;PRINT HL VAL ON SCREEN
BBB2 EB                  EX      DE,HL           ;SAVE HL REG.
BBB3 CD25BB              CALL    PUSHAL          ;SAVE REGS.
BBB6 EB                  EX      DE,HL           ;RECOVER ORIG HL VALUE
BBB7 CD9A0A              CALL    HLACUM          ;HL==> ACCUMULATOR
BBBA CDCC0A              CALL    ACINSN          ;ACCUM INT=> SINGLE PREC.
BBBD CDBD0F              CALL    ACUSTR          ;MAKE ACCUM INTO STRING
BBC0 23                  INC     HL              ;SKIP LEADING SPACE
BBC1 CD752B              CALL    PRINT           ;PRINT VALUE
BBC4 CD2EBB              CALL    POPAL           ;RESTORE REGS.
BBC7 C9                  RET
                ;
                ;
                ; Now table, message, & buffers
                ;
BBC8 56         MSG      DEFM    'Variable Name or <ENTER> '   ;*
BBE1 00                  DEFB    00H             ;END OF MSG MARKER
                ;
BBE2 25         TYPES    DEFB    '%'             ;LIST OF VARIABLE TYPE
BBE3 24                  DEFB    '$'             ;  MARKERS
BBE4 21                  DEFB    '!'
BBE5 0000                DEFW    00H             ;PAD WITH 3 SPACES
BBE7 00                  DEFB    00H
BBE8 23                  DEFB    '#'
                ;
0014            VARBUF   DEFS    20D             ;*BUFFER FOR VAR. NAME
                ;
0400            SCRBUF   DEFS    400H            ;BUFFER FOR SCREEN
                ;
0002            CRSBUF   DEFS    2H              ;BUFFER FOR CURSOR POSN
                ;
BFFF FF         FLAG     DEFB    0FFH            ;*RECURSION FLAG
                ;
B9C5                     END     SETUP           ;END OF PROGRAM
00000 Total Errors
```

```
LOOP    LD      A,(3BFFH)   ;LOOK FOR KEYSTROKE
        OR      A           ;SET FLAGS
        JR      Z,LOOP      ;LOOP UNTIL STROKE FOUND
        LD      A,R         ;ELSE GET R VALUE
        AND     7FH         ;MASK OUT BIT 7
                            ;A HAS RANDOM NUMBER
```

Program Listing 2. Trigger routine for reading the R register.

them as it uses them, you can define simple variables the same way you define arrays: with the DIM command. DIM A,B,C defines, and sets up space for, the three variables listed.

Therefore, a sensible order of commands at the beginning of a program would be:

- Clear sufficient string space;
- Define or Dimension simple variables;
- Dimension array variables.

The second lesson to be gained from variables displays is that the order of variables in memory makes a difference. Whenever Basic has to deal with any variable, it searches the appropriate table from the beginning. If it finds the variable, it continues to process the current command. If it can't find the variable, it (usually) must make room for it and add it to the table.

If your most frequently used variables are at the bottom of the table, they will be found faster and your program will run more quickly. To place them at the bottom of the table, be sure to define them first.

One interesting side note—if you ask Basic for the value of a simple variable that you have not previously defined, it searches the SVT and, failing to find the variable, answers zero for a numeric variable or " " for a string variable. However, it will *not* add the variable to the SVT.

If you ask Basic for the value of an element in an array that you haven't previously defined, Basic first creates the array in the AVT, using the default value of 10 for the maximum of each index, and then reports that the value of the array element is zero or " ". Be careful when you use arrays; they can fill up memory quickly. The innocuous looking statement:

```
DIM A#(7,8,8,8)
```

requires 46,666 bytes, and uses up almost all available memory in a 48K tape-based Model III.

Random Numbers

In my own programming and in questions from readers, the subject of random numbers has come up several times. There seems to be a misunderstanding about what a random number is and how random numbers are generated internally in Basic.

First, generating a random number is simple on a Z80-based machine. The Z80 uses the R register internally to signal memory refresh cycles. During the decoding and execution of every machine-language instruction, Z80 increments the R register, places it on the address bus, and generates a memory refresh signal. Since this occurs while the Z80 is processing a program instruction, it takes no extra execution time and is generally transparent to the running program.

If, at an unpredictable time, the Z80 reads the R register, its current value is a random number. For the read's time to be truly unpredictable, some event must trigger it, that, even if repeated frequently, would take much longer than several machine cycles to occur. One such triggering event might be a key pressed by the computer operator.

Try the following experiment to see how the computer generates random numbers. Take a digital watch that has a stopwatch function and can record hundredths of a second. Start the watch, wait a moment, and then press stop. Record the digit in the hundredths column. Now try to repeat the experiment and stop the watch with the same value in the hundredths column. Slow human reflexes combined with the stopwatch button's resistance make the digital stopwatch a reasonable random number generator for numbers between zero and 9 in the hundredth's column. No one could purposely stop the watch with the same digit showing each time.

The Z80 updates the R register much more frequently than the relatively slow hundredths of a second display on a digital watch. Because bit 7 of the R register never changes (you can set that bit by loading a value into R), reading the R register at random times produces a truly random number between zero and

"The Z80 updates the R register much more frequently than the... display on a digital watch."

127. By pressing a key, you can trigger a read, and the code would look something like Program Listing 2.

When you execute the Basic command Random, it also uses the R register, but in a different manner. The entire code for RANDOM is:

```
LD    A,R          ;GET CURRENT R
                    VALUE
LD    (40ABH),A    ;SAVE IT
RET                ;END OF ROUTINE
```

The exact instant when Random is executed is certainly unpredictable: It depends on your loading a program that contains the Random command, typing RUN, and pressing the enter key. The only mystery in those three lines of code is why the value is stored in 40AB hex.

The RND function in Basic produces pseudo-random numbers. The values produced seem random, but you could easily predict the next random number if you knew the present state of the computer. Pseudo-random numbers are generated by taking a "seed" value and performing a specific series of arithmetic operations on that value. The new seed value becomes the new pseudo-random number.

The computer stores the random number seed in both the Models I and III at memory locations 40AA hex, 40AB hex, and 40AC hex. The specific algorithm used to generate each successive seed is unimportant here; what is important is that if you know the current seed value and the algorithm, you can accurately predict the next pseudo-random number. In some types of modeling, it's important to be able to use the same set of random numbers repeatedly in a program to test various hypotheses. To see how such a program might operate, run the following:

```
 10 FOR I=1 TO 5
 20   POKE 16554,1
 30   POKE 16555,2
 40   POKE 16556,3
 50   FOR J=1 TO 8
 60     PRINT RND(100),
 70   NEXT J
 80   PRINT
 90 NEXT I
100 GOTO 100
```

Lines 20, 30, and 40 establish the current seed value, thus determining the series of pseudo-random numbers generated.

It should be clear now how the Random command operates. It takes a truly random value—the current value in R—and uses it to re-seed the pseudo-random number generator. The software pseudo-random number generator always generates the same series of values; the computer uses the Random command to start that series at a random spot on the list. The total list of pseudo-random numbers is long enough that you will probably never have a program that will notice a repetition of values.

If you wish to generate your own pseudo-random numbers in machine language, you'll need to do some research about different pseudo-random algorithms. You will find a great deal of disagreement among the experts about which algorithm is best, and what constitutes a truly random pseudo-random number (if such a beast exists). The ideas are interesting, but the mechanics soon become extremely complex.

Authors' Forum

As I mentioned last month, readers who subscribe to CompuServe may take part in open discussions of topics covered by "The Next Step." GO PCS-117 to the Software and Authors Special Interest Group (SASIG) and leave your questions or comments addressed to me on the message board. Feel free to join in any discussions started by other readers. ■

Write to Hardin Brothers at 280 N. Campus Ave., Upland, CA 91786, or contact him through CompuServe. His e-mail address is 72165,735.

✓ See List of Advertisers on Page 291

by Scott Norman

Final words on the CoCo

I've got good news and bad news. The bad news is that this will be the last edition of The Color Key; the good news is that next month, I'll be taking over the Re:FLEX column in *HOT CoCo* and expanding its coverage to include both FLEX and OS-9 software.

While most of the previous action in the area of alternative operating systems centered on FLEX, every indication is that the long-delayed OS-9 software will have a broad range of supporting applications programs for the serious user.

I recently spent an afternoon with Sue and Paul Searby of Computerware, discussing some of their plans for new products. It comes as no surprise that they will market a rather complete line of OS-9 business software, and you can bet that other suppliers will do the same.

Since that's where much of my interest lies, I'm looking forward to working with and writing about the new system. I expect to be in the thick of it by the time you read this column.

And by the way, the fact that Radio Shack is advertising Basic09 for the Color Computer is the best news to come out of Fort Worth in a while. I can hardly wait.

Unfinished Business

Some bugs are subtle and some are just plain dumb, and one of the latter kind has surfaced in my Expgraph program (September 1983, p. 30). Let's see if I can make amends.

You might recall that Expgraph creates a high-resolution graph of expenditures as a function of time for periods of up to a year. The program automatically scales the dollar axis to maximize the resolution of the graph, while restricting the major interval to an integral multiple of $2, $5, or $10 times some power of 10. It also extrapolates the rate at which you spend money, so the user can compare the prediction for year's end with a budget target.

The bug appears in the routine that converts the value used for the expenditure-axis interval to a string in preparation for drawing it on the high-res screen. As I learned when working with one particular set of project cost figures, I should have included code to force this interval to be an integer.

As things stand, the program is capable of arriving at an interval of, say, $50000.01. This in turn would appear on the graph, sans decimal point, as a scale factor: $5000001. Very sloppy, to say nothing of downright wrong.

The solution is simple enough. Change the first half of Expgraph line 1140 from

```
C$=STR$(C)
```

to

```
C$=STR$(INT(C))
```

to enjoy classy displays.

Sorry about that.

What Do You *Really* Use?

A number of people have written with similar questions: How can I possibly use all the software I review? Why does anyone need 15 data file managers, half a dozen word processors, and three or four spreadsheets? In fact, do I use the stuff at all, or do I give it a quick once-over before consigning it to the wastebasket?

Fair enough. I confess that until I began to get some reader feedback, it never occurred to me that people would think everything I reviewed favorably I'd automatically add to my own collection of everyday tools. I should be flattered, I guess.

Actually, my correspondents are quite right; limits exist to what anyone really needs, and limits to what anyone can profitably use. It makes little sense to spend all your time learning new command sequences so you can have the latest wrinkle in a particular type of program—unless you need that wrinkle. Therefore, my standard software library changes fairly slowly.

At the same time, I think it's incumbent on me to thoroughly wring out the products I review. I try to spend enough time with each program to explore all its major features.

That takes a fair amount of time, and occasionally it isn't possible to exercise every option of a complex product. Naturally, when reviewing software I only report on the features and commands that I have actually used.

I rely on a fairly standardized set of procedures to test the major types of programs: word processors, file or data-base managers, spreadsheets, and so forth. For example, I test spreadsheets with some dummy research and development (R&D) project budgets, departmental salary plans, and IRS forms. These simulate my principal real-world applications, and give me an opportunity to see how each new review subject handles a typical set of operations.

I often have to depart from my routine to explore novel features of a program, though, and such explorations sometimes convince me to add a product to my stable.

That's how I decided to start using Derringer Software's Pro-Color-File (P.O. Box 5300, Florence, SC 29502) for my heavy-duty data file management. Its particular capabilities for computation and report generation address some of my special needs, so I thought it worth converting many of my files to Pro-Color-File format.

This involves a fair amount of effort, and I certainly wouldn't recommend that everyone start from scratch whenever a new program shows up. My point is that in trying to keep my reviews honest, I have the opportunity to be tempted to buy more software than the typical user. As a result, I probably get involved in more file modification and rewriting than a sane person would tolerate.

I don't always change every file over to accommodate a new product, since I

have some duplication in my working collection.

Now I'll answer the question I raised as the title of this section. Please remember the strong personal component in this business before you write to squawk about my choices. I've favorably reviewed plenty of software that failed to make me switch for my own use.

I have used Telewriter (Cognitec, 704 Nob Ave., Del Mar, CA 92014) for the bulk of my word processing ever since its inception. When the current version, Telewriter-64, became available, its ability to right-justify text caused me to abandon my second word processor.

"CoCo users are lucky where spreadsheets are concerned."

I previously used Trans Tek's C.C. Writer (194 Lockwood Lane, Bloomington, IL 60108) for anything resembling formal correspondence, but now I had no motivation for continuing with two programs for one job.

Telewriter's new-found ability to produce ASCII files doesn't hurt, either. Now it's much easier to use auxiliary programs such as spelling checkers; I use Spell 'N Fix (Star-Kits, P.O. Box 209, Mt. Kisco, NY 10549). I can read and edit files generated by some other programs, too.

The whole topic of file compatibility between programs is something of a sore spot with me, incidentally, and I'll return to it later.

I've already mentioned using Pro-Color-File for heavy file management. This is an area in which I indulge myself a little; I have two other file managers in my household.

I still use Trans Tek's C.C. File for casual, unstructured applications like my file of addresses and phone numbers of CoCo vendors. Some of the entries include reminders of what the vendor produces, others do not, and the whole thing is gloriously loose. Ditto for my family's file of favorite restaurants. C.C. File's lack of structure and limited command set are especially welcome since I call on it sporadically without reference to the documentation.

I also have specific applications for Radio Shack's Color File. For some time, I have used it to maintain a text file I call COCOFACTS, which contains notes about various aspects of Color Computer operations.

This is where I keep track of all the PEEKs and POKEs I use in programming. How else would I remember where the Next-Data-Item pointer is? Color File's two-level indexing scheme and on-screen prompts seem right for these applications.

In principle, I could probably handle all my file management chores with one program. I have used Homebase (Homebase Computer Systems, P.O. Box 3448, Durham, NC 27702) and can vouch for its ability to handle data and text files.

The trouble is, I've become comfortable with my existing three-tier system and don't feel like adding a fourth component or translating all my data files. The urge I had to unify my word processing is lacking here.

CoCo users are lucky where spreadsheets are concerned. For a while the Spectaculators, disk and ROM pack, were pretty much the whole story, and their capabilities were relatively limited. Then Trans Tek's C.C. Calc acted more like VisiCalc *et al.*, but was hampered by being written in Basic.

Then this spring Elite Calc (Elite Software, Box 11224, Pittsburgh, PA 15238) became available, and suddenly unmodified CoCos have a full-blown spreadsheet. In the meantime, FLEX users could call on the big-league power of Dynacalc (Computer Systems Center, 13461 Olive Blvd., Chesterfield, MO 63017).

I've used all four, and still have both Elite Calc and Dynacalc files in abundance. I could happily switch to 100 percent use of either program, but other considerations (see the end of this column) make me want to stay sharp on both. That's not too tough; Elite Calc has a simple command syntax, while Dynacalc's resembles that of SuperCalc, the CP/M-based spreadsheet I use regularly at my office.

Those are my big guns—the programs I use almost daily. Of course, a lot of utilities come into play for special purposes: Master Control (Soft Sector Marketing, 6250 Middlebelt, Garden City, MI 48135) and Colorkit (Arizin, P.O. Box 8825, Scottsdale, AZ 85252) to name two.

I've also been experimenting with several math and business graphics packages and expect to become a steady user of one or more, but it's early to declare my allegiance. My reviews have been appearing here and in *HOT CoCo* on a fairly regular basis; look for more on graphics over the next two or three months.

Odds and Ends

I'd like to return to the question of the Color Computer versus other serious 8-bit machines. The CoCo programs I use (and many of their major competitors) are the equals of the better-known applications programs for other computers. I use CP/M software almost daily, but that doesn't mean it overwhelms me.

Both Elite Calc and Dynacalc are better in some respects than early versions of Supercalc, which is itself arguably better than VisiCalc. It has taken the newer Supercalc 2 to incorporate some features of the CoCo programs: the ability to sort a spreadsheet by row or column data, for example.

Telewriter-64 is far easier to use than WordStar, and does a perfectly satisfactory job on anything but the most specialized text processing. For that matter, Super Color Writer (Nelson Software Systems, 9072 Lyndale Ave. South, Minneapolis, MN 55420) can take on WordStar, too.

As for file and data-base managers, I haven't done enough work with CP/M material to form a definite opinion. I know that dBase II is very powerful but quite complex—more complex than Pro-Color-File, Homebase, or several other top CoCo file managers. The CP/M program might do all kinds of exotic tricks, but for the moment I feel that CoCo users have some competitive software at their command.

What troubles me, though, is that it's a real chore to use the output of one program as the input to another. Life would be a lot simpler if I could pop an Elite Calc data file into Radio Shack's Disk Graphics to produce a bar chart, or if I could get such a chart into a Telewriter report.

Of course it's possible to write a conversion program for almost any specific application, but wouldn't it be nice if CoCo programs talked to one another with less fuss?

Some CP/M software does better: You can use WordStar with auxiliary

✓ See List of Advertisers on Page 291

programs to prepare indices and footnotes, for example. I don't think this desire for well-integrated software is a fetish of mine alone. The best feature of Apple's Lisa is its elaborate suite of business programs, and integrated packages like 1-2-3 promise to be huge sellers in the IBM PC community.

Clearly, someone else cares about this problem. The existence of a standardized operating system helps, and maybe the official blessing of OS-9 will result in better-integrated software for Color Computers.

Super CoCo

The 64K Color Computer and the Color Computer 2 are now appearing in Radio Shack catalogs, and the first samples of the Dragon 32 are out. The former and the latter have some appealing features, but neither is perfect.

Therefore, I'd like to share a few fantasies with you and set down some of my thoughts for a dream machine. This isn't a truly radical proposal, just some ideas I'd like to see incorporated into a next-generation Color Computer.

I'll start with a baseline machine having 64K of RAM and the best of the full-travel keyboards. Now, how about adding an official clock speedup to 1.8 MHz. Maybe the operating system would have to shift in and out of high gear for I/O, but this shouldn't concern the user. Add an internal muffin fan if necessary to preserve IC lifetimes.

The machine needs a few user-definable keys like the ones on the Model 100 and some of today's add-on keyboards. I hope software vendors would prepare patch areas so you could add customized definitions for such keys to their programs. One of the things WordStar does right is allow control/digit key to represent a series of keystrokes, making it much easier to enter frequently used command sequences.

"I'd like to share...my thoughts for a dream machine."

A baseband video output for use with a monitor could be a monochrome signal, and would be used primarily for word processing and an 80-column spreadsheet. The Dragon 32 has baseband and RF outputs already.

I'd like better A/D converters. Eight-bit resolution would be nice, to increase today's 64 resolvable analog input levels to 256. An integral Centronics parallel port should go along with the RS-232. Let's get serious about printing, and do away with the external boxes and PC boards.

I want gold-plated cable connectors everywhere. Weak links like the CoCo's disk controller connectors aren't charming idiosyncrasies; they're embarrassing flaws.

Dedicate a second 6809, with perhaps another 64K of its own RAM, to control the display. I'm not kidding; at least one such machine already exists in Japan. It would be great to have this kind of power for graphics of all kinds. Consider the possibilities of 40 PMODE 4 pages, for starters.

One of the nicest things about designing dream machines is that you don't have to worry about cost.

The End

It's the end of the road, ladies and gentlemen. I've enjoyed writing The Color Key, and I hope that many of you will find something of interest in the new FLEX/OS-9 applications column.

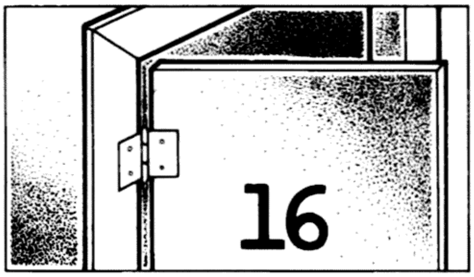
I don't intend to abandon the rest of the CoCo world. One of my major current projects is a book in which I'll treat applications software in more detail than the column/review format permits.

With any luck at all, the book will be out in the autumn of 1984. The publisher is Scott, Foresman.

In the meanwhile, I'll see you in *HOT CoCo.* ■

Contact Scott Norman c/o The Color Key, 80 Micro, *80 Pine St., Peterborough, NH 03458.*

See List of Advertisers on Page 291



Arcnet and Xenix hubbub

Perhaps the most powerful asset of any 16-bit microcomputer is its ability to become an office or interoffice hub that other terminals can access. Its multi-user and multi-tasking capabilities blur the fine line that separates minicomputers from microcomputers.

Terminals, commonly referred to as work stations, are typically less expensive than complete computer units. Using several terminals to tie into one host computer allows many people to take advantage of the main computer simultaneously.

The host computer, in its standard configuration, has a hard disk drive connected to it. All of the terminals can access information stored on the same hard disk.

Currently, Radio Shack offers two ways to add external remote work stations to the Model 16. These networking systems are called Arcnet and Xenix.

Arcnet

Arcnet, an acronym for attached resource computer network, lets you connect up to 255 computers to the Model 16.

The terminals can use all of the peripherals connected to the host machine. These include both floppy and hard disk drives and printers. You cannot use the host machine, called a file processor, as a terminal. Its keyboard and video display aren't available to the operator.

Arcnet also runs on the Models II and 12. It doesn't require the MC68000 board, but every computer in the system must have an Arcnet circuit board installed. Have your local Computer Center do this.

Hubs, similar in function to a TV antenna splitter, join the work stations and the host machine together. One cable connects to a hub's input and provides several outputs to feed other work stations. Passive hubs let you connect together four units up to 200 feet from the hub.

Active hubs boost the data signals with internal circuitry so you can place terminals as far away as 2,000 feet from the hub. Combining many active and passive hubs together yields a maximum distance of four miles between the farthest work station and the host computer.

Obviously, Arcnet is a good choice for multi-user applications where work stations are located in different buildings (such as a college campus).

The owner of multi-user work stations must run the necessary cable wires through the buildings. Radio Shack computer technicians help with the installation of their computers in an Arcnet system but you must get an electrician to prepare the necessary wiring.

We haven't had the opportunity to work with an Arcnet system and welcome comments from readers who use one. Our understanding is that it operates at a speed of 2.5 million bits per second. With that kind of speed you feel as if you are the only user on the host computer.

Xenix

The second multi-user system currently available is TRS-Xenix, or simply Xenix. In contrast to Arcnet, which is a hardware configuration, Xenix is software based and requires no special circuit modifications on any of the system's microcomputers. Since it uses the MC68000 microprocessor, you need a Model 16, 16B, II/16, or 12/16 to run it.

Unlike Arcnet, the host computer's video and keyboard are available as a work station. However, you can connect only three terminals (including the host) at one time. The other one or two terminals interface by way of the two RS-232 jacks on the back of the host computer. A null modem adaptor does the job along with RS-232 cables.

Under Arcnet, each work station is a complete computer, not just a data terminal. Xenix requires that only the host machine be an independent computer.

As of this writing, you need a hard disk drive to run the system, but a floppy

Assembly-Language Corner

Prior to displaying any information on a video screen, it's usually necessary to clear the screen and position the cursor at a point where you want to begin printing. This month we take a look at some of the supervisor routines available to perform these screen formatting functions.

The video character generator circuit in the Model 16 is identical to that of the Models II and 12. Therefore, all graphics characters and ASCII codes are compatible.

Normally the screen format of these computers gives us 80 character positions horizontally and 24 vertically. The video generator is capable of printing characters in a larger mode. While the vertical count remains 24, you can double the width of each letter. Thus, in the large character mode, a maximum of 40 letters fits on a horizontal line.

Also at our disposal is an inverse

A-L Corner continued on next page

version is in the works. Naturally such a version is limited due to less disk space on a floppy.

Xenix's basic structure builds around Western Electric's popular and established Unix operating system. Xenix comes from a thoroughly tested program in the field for 10 years.

Since more than one person can work on the same disk file at the same time, imagine the disaster if two people try to write the same record to a file simultaneously. Fortunately, Xenix designers took this into consideration. The program doesn't allow two users to write information to the same record and accidentally lose data.

Xenix divides the computer's RAM into separate sections for each user. In this way, each of the two or three users can run programs independently. One can run payroll while another works on accounts receivable, for instance. Xenix runs with a minimum amount of 256K RAM. However, certain combinations of programs run simultaneously may require 384K or even 512K.

Both Arcnet and Xenix have their pros and cons. Your choice depends on your business's particular needs. Xenix doesn't require you to make any hardware modifications to existing computers; Arcnet does. But Arcnet handles up to 255 computers; Xenix accommodates only three. With Xenix, the remote work stations need only be data terminals such as Radio Shack's Model DT-1. Under Arcnet, each work station must be a computer. ■

A-L Corner continued from previous page

video option. Normally the background of the screen is unlit or black and the letters light up (green or white, depending on your machine). You can reverse this to cause the background surrounding a letter to light up and the letter itself to appear as a darkened area within the block.

Built within the disk operating system is a routine (referred to as a supervisor call) that you can use to establish the size of the letters and the normal/inverse printing font.

Placing a zero into byte-offsets 6 and 7 of the SVC block (a buffer area you set up to pass values to the DOS routine) switches the video size to the 40-character-per-line mode. A value of 1 placed there produces 80-character lines. In byte-offsets 8 and 9, a zero indicates inverse video and a 1 indicates normal printing.

The supervisor number that identifies this routine from other supervisor calls is 7. Always place the identifying supervisor number in byte-offset zero of the buffer. The set-up to call this routine looks something like this:

```
  LDA      .A0,SVC BLOCK
  MOVW     @A0,#7
  MOVW     6@A0,#1
  MOVW     8@A0,#1
  BRK      #0
  RET
SVC BLOCK
  RDATAB   32,0
```

Use the move-a-word (movw) command to load the necessary values into the buffer area. This is an indirect addressing mode where register A0 stores the address pointing to the location of the SVC block buffer.

With the values you use in the example, prepare a normal screen format—80 characters per line and no inverse video.

Supervisor call 7 automatically performs two other functions. It clears the screen (similar to the Basic CLS command) and it homes the cursor—moving it to the top leftmost position on the screen.

You can use another supervisor routine to clear the screen. This is call number 8 which sends a character to the video display. Examine the machine's ASCII code chart in the owner's manual and note that the decimal number 30 is a control code for clearing the screen. By sending this ASCII code to the routine that prints a character on the display, you can clear the screen. However, unlike the last routine, this doesn't set up the inverse/normal and 80/40 screen formats.

```
  LDA      .A0,SVC BLOCK
  MOVW     @A0,#8
  MOVW     6@A0,#30
  BRK      #0
  RET
SVC BLOCK
  RDATAB   32,0
```

Positioning the Cursor

Before printing any letters or graphics characters on the video display, you may want to position the cursor at a specific point to start printing. Supervisor call number 10 lets you place the cursor at any printable location on the screen.

You must move values for the horizontal (*x*) and vertical (*y*) coordinates into the SVC buffer area to instruct the computer as to the row and column on which you desire to place the cursor.

Place the value for the row position in byte-offsets 6 and 7. The row position refers to the number of lines down from the top of the screen. Offsets 8 and 9 store the column position. This is the number of character positions from the left-hand side of the screen.

Since there are 24 lines down the screen and 80 positions across, halving these values to 12 and 40 places the cursor in the center of the screen.

```
  LDA      .A0,SVC BLOCK
  MOVW     @A0,#10
  MOVW     6@A0,#12
  MOVW     8@A0,#40
  BRK      #0
  RET
SVC BLOCK
  RDATAB   32,0
```

edited by Lynne M. Nadeau

Figure 1. Monty and Eric play Scrabble.

Review Contents

Our reviewers use a five-star rating system. One star represents the low end of this spectrum, while five stars represent the spectacular and high end of the spectrum.

★★★★

Monty Plays Scrabble
Tandy/Radio Shack
One Tandy Center
Fort Worth, TX 76102
Model III, 32K
One disk drive
Back-up limited
$34.95

by Eric Maloney
***80 Micro* staff**

Monty Plays Scrabble isn't going to win any tournaments. But it's good enough to give the average player a run for his money, and challenging enough to give the experienced player a stimulating practice partner.

To play Monty, you need a Scrabble board and a good dictionary. Either you or the game can pick the tiles. The program provides a board on the screen, showing you where it placed its word and letting you indicate where you want yours.

Among other features (see Table 1), Monty gives you four skill levels, keeps track of the score, and lets you save a game in progress. A game takes about one hour and 40 minutes to play.

But enough detail. What kind of Scrabble does Monty play?

At first, I was disappointed. I won my first game by the depressing score of 468 to 288, and took the next five as well. The average tally was 431 to 302. This, I thought, is no way for a computer with a 54,000-word vocabulary to perform.

But Monty surprised me in game seven, reeling off three seven-letter words en route to a 440–383 win. And while it hasn't beaten me since, it has occasionally given me a game worthy of a capable human partner.

A Typical Game

The best way to demonstrate Monty's abilities is to recount an actual game. This one is our eighth, and is fairly typical. We play at the highest (Scholar) level, and Monty goes first. Figure 1 shows the final board.

1. My letters: FTAODRT. Monty starts off with CAY, whatever that is. I counter with FART. (So who says Scrabble has to be polite?) "Good play!" Monty responds. He says that a lot, even for words like IT. Score after one turn: 16–26, my lead.

2. Letters: *IINODT. Some good possibilities here—DICTION comes immediately to mind. I wait patiently while Monty thinks; it takes him about three minutes of disk I/O to make a move.

After much grinding and gnashing, he plays MOAN, MAY, OR, and AT for 22 points. Not too bad. Since I have no place to put DICTION, I play DICTATION instead for a quick 65. "Very great word!" Monty exudes. Monty might have an elephantine memory, but he has yet to learn how to use adverbs well. Score: 38–91.

3. Letters: ETEEOID. Monty's revenge. I need to get rid of some of these E's. Too bad EDICTATION isn't a word. What's taking Monty so long? I could spin off a game of Galaxy Invasion in the time he needs to think.

"Aha!" he exclaims. That must mean he's done. He lays down GAP, AD, and PI for 19, and opens up the triple word score. Unfortunately, I can't use it for anything. I play DECEIT and GAPE for 23. Score: 57–114.

4. Letters: RHQPOOE. Stuck with the Q—and with no U in sight. "I'll be ready soon," says Monty. I've heard that line before. Think I'll go change my oil. He finally plays BUM, BO, and UN. Wait a minute—UN??? Monty's played some weird words, but this one's a bit too much. Seeing, however, as I don't have an official Scrabble dictionary, I let it go.

Figuring that two can play the weird word game, I put down HOOPER, HUN, and OM. But wait—Monty challenges OM! Is it a word? he asks. I look it up in my Webster's, and sure enough, there it is. "Monty regrets his error," he says. Score: 76–160.

5. Letters: IEARGDQ. Monty loses his turn for an unsuccessful challenge.

Taking full advantage, I play AGED, BOA, and HUNG for 25. Score: 76–185.

6. Letters: IEARGLQ. This rack looks suspiciously familiar. Monty starts thinking. I go out for dinner and a drink. I return to find that Monty has played BUNG. Isn't he a character in The Wizard of Id? I play GLARE. Score: 97–207.

7. Letters: OKSVXQI. Talk about a constipated rack. Monty plays TOWEL for 26. I counter with XI, XI, and IT for 31. XI is one of my favorite Scrabble words, and I'm delighted to be able to play it twice on one move. Score: 123–238.

8. Letters: OKSVTQC. It's getting worse. Monty plays IDS and XIS for 18. I put down LOCK for 30. Vowels! I need vowels! Score: 141–268.

9. Letters: ERSVTQE. That damned Q! Monty plays RUIN for 15, I can't take it any more and exchange my Q, getting an A in return. With my luck, I'll pick up the Q again later. Score: 156–268.

10. Letters: ERSVTAE. Monty plays YARNS for 24. Getting rid of the Q pays off—I play AVERTERS for 62. Score: 180–330.

11. Letters: OLIIEQW. There it is again. Monty plays JEE for 26. I play WILIER for 18. Score: 206–348.

12. Letters: VIZNOOQ. I shout at the Q to stop torturing me. Monty dives into the well of contemplation; I go reshingle the house. He finally puts down HATSFUL for 26. He picks up the remaining tiles, thus sticking me with the Q for all eternity. I play AZO for 32. Score: 232–380.

13. Letters: NVIOQ. Monty plays EASE, ETA, DOS, and WE. DOS! This is too much. I play VIOL for 14. Score: 257–394.

14. Letters: QN. The end is near. Monty goes through his usual gastrointestinal tremors and lays down VI*A for 12 points. That, as they say, is that.

1	Exchange tiles
2	Pass or forfeit
3	Save game to disk
4	Current totals
5	Check your tiles
6	Change skill level
7	End the game

Table 1. Monty plays scrabble options.

Final score: 280–383.

Final Comments

Clearly, a huge vocabulary doth not a Scrabble player make. You need to be a good strategist, too. Monty is not; it is apparently programmed to go for the highest possible point total, whether that means throwing away an S, breaking up a potential seven-letter word, or opening a triple-word score for its opponent.

Monty wins an occasional game, but it is through brute force rather than cunning.

Still, you don't need a great opponent to exercise your own Scrabble skills. Monty gives you enough of a challenge to keep you from getting bored. And it sends you to your dictionary enough times to increase your own vocabulary. In recent games, it has spun off such words as indium, llano, uta, rabbet, vug, aff, eme, and dommir.

One final note: Monty Plays Scrabble allows you only one back-up. This is a serious problem with a program that accesses the disk some 200 times per game. Monty could have a short life if you don't figure out a way to break the protection scheme and give yourself a full supply of copies. ■

★★★★½
The Statistician
Quant Systems
P.O. Box 628
Charleston, SC 29402
Models I, II, and III
$125

by John Dunkelberg

The Statistician is an excellent statistical and forecasting system that contains a wide variety of simple and sophisticated statistical analyses. The program is especially useful for business forecasting and complex statistical analysis. The Statistician is also an excellent tool for an instructor in an elementary or advanced statistics course.

The Statistician is completely menu driven so it's easy to use, even for a microcomputer novice. Thirty minutes after reading the documentation, I was running my first regression.

Multiple Regression

One of my principal reasons for acquiring The Statistician was the multiple regression package. The Statistician contains five different regression procedures including Stepwise, Ridge, Backward Elimination, and All Subsets regressions. I haven't seen Ridge, Backward Elimination, or All Subsets procedures in any other statistical package for micros.

You can create large models with up to about 50 independent variables. This is enormous, especially when compared to the Radio Shack statistical analysis program that only allows five independent variables.

The output (see Table 1) closely resembles that of mainframe packages and includes t values for the individual coefficients as well as their standard errors. Also, the program includes the Durbin Watson statistic, which is useful in residual analysis and is found on few other statistical programs for microcomputers.

You can list or print the variance/covariance and sums of squares matrix as well as the correlation matrix of the estimates. Residual analysis is also good. You plot residuals or list them with the actual and predicted values on the screen or printer.

Another feature I like is that the program saves the predicted values to disk. This lets you estimate simultaneous equations models. In addition, the Sort utility ranks the residuals by actual or predicted values of the independent variable.

I tested The Statistician's accuracy on the Longley data, a benchmark for testing statistical accuracy. I found The Statistician superior to the mainframe programs tested by Longley in 1967. The program's accuracy was amazing.

Data

The Statistician provides an easy data entry and editing system. You specify the number of variables, then the program displays the appropriate number of fields on the display. By pushing the appropriate arrow keys, you move around the fields or up and down through the rows of data.

All files that the editor writes out

become part of your data base and any of the programs in the package can use them. The edit program contains a good data report formatting system that automatically centers titles and headings and aligns data.

One of the most important components of the package is the data transformation function. It takes some time to become accustomed to this particular program, but it's worth it. Using any of the 24 transformations, you modify data or create new variables as functions of existing variables.

The transformation capability is essential for non-linear multiple regression and many techniques in exploratory data analysis. Using this feature, you can easily create the necessary variables for a polynomial or interaction regression model.

Some of the data transforms are quite unusual but occasionally useful: for example, the additive and multiplicative accumulators. These accumulators calculate the cumulative sum and product of a vector.

Some of the transformations are designed for time series modeling. In particular, the *n*th order lag lets you create lagged data of any specified order.

Other Features

The descriptive statistics component computes the following numerical descriptive measures: mean, median, geometric mean, harmonic mean, variance, standard deviation, maximum and minimum values, mean absolute deviation, and range. In addition, the program produces excellent frequency histograms (see Fig. 1).

DEPENDENT VARIABLE-->PRICE/DAT

		COEFFICIENT	STD. ERR.	T-VALUE
CONSTANT	B 0	33482	32100.7	1.04303
FEET/DAT	B 1	-30.2504	21.2985	-1.42031
AGE/DAT	B 2	-608.578	450.463	-1.35101
BEDS/DAT	B 3	17597.7	7304.79	2.40906
BATHS/DAT	B 4	27221.1	6765.33	4.02362
GARAGE/DAT	B 5	-5487.06	6052.39	-.906593
RESORT/DAT	B 6	40420.5	7913.66	5.10769

SOURCES OF VARIATION	ANOVA SS	DF	MSE
REGRESSION	1.67561E+10	6	2.79268E+09
ERROR	5.5722E+08	8	6.96525E+07
TOTAL SS	1.73133E+10	14	

F= 40.0945
R SQUARED= .967816
DURBIN WATSON STAT. = 2.35218

Table 1. The Statistician's output.

```
1 LINE = 1 OBSERVATION(S)

 14   :                          ***********
 13   :                          *         *
 12   :                          *         *
 11   :                          *         ***********
 10   :                ***********         *         *
  9   :                *         *         *         *
  8   :                *         *         *         *
  7   :                *         *         *         *
  6   :      ***********         *         *         ***********
  5   :      *         *         *         *         *         *
  4   :      *         *         *         *         *         *
  3   :      *         *         *         *         *         ***********
  2   :      *         *         *         *         *         *         *
  1   :      *         *         *         *         *         *         *
  0   :      *---------*---------*---------*---------*---------*---------*----
              < 9         11        15        19        23      > 25
```

Figure 1. Frequency histogram.

The program also permits several types of hypothesis testing, including tests on a single mean, single variance, and difference in means (paired and unpaired). The paired difference test uses the differencing functions in the transform package, then performs a test of a single mean on the differences. One-way ANOVA measures the difference in unpaired means.

The ANOVA provides one- and two-way analysis of variance; however, the program doesn't support multiple observations per cell in the two-way analysis.

The nonparametric component contains six commonly used nonparametric tests: Median, Mann-Whitney, Wilcoxon, Kruskal-Wallis, Spearman's Rho, and the Runs Test.

The forecasting program is one of the strongest available. It performs no less than eight different time series models, including moving averages, single and double exponential smoothing, sinusoidal models, Holt's two-parameter linear exponential smoothing, Winter's exponential smoothing model, and adaptive filtering.

It calculates the mean squared error and the mean absolute deviation of the forecast. The Statistician also produces tables and plots of the predicted, actual, and forecasted values on the screen or printer.

The program generates random variables from seven different types of distributions: Normal, Gamma, Exponential, Uniform, Poisson, Binomial, and Geometric.

Another component that particularly applies to teachers and students of statistics is the sampling program. You can get sampling distributions from any data set. The program obtains all random samples of a given size if you have sufficient disk space on your system.

The documentation is well-written and tutorial in nature. Since the system is menu-driven, it's more than adequate. I quickly obtained answers to my few questions with a telephone call.

Conclusion

A second version of The Statistician with enhancements is due for release in September. The new version will cost $295, and a regression subset will be available for $145.

The Statistician is an excellent program for anyone interested in performing statistical analysis. Moreover, if you've already purchased statistical software, you certainly should consider acquiring The Statistician; it contains many features that just aren't available in any other package. ■

★★★★

Newbasic 2.1
Modular Software Associates
209 18th St.
Huntington Beach, CA 92684
Models I, III, and 4 (in Model III mode)
$39.95

by Richard Green

Newbasic 2.1 is an enhancement to Disk Basic for the TRS-80 Models I, III, and 4. Newbasic includes 49 new Basic commands. You can add any or all of them to Basic at your discretion.

The program comes on a 35-track, single-density Model I disk. This disk contains programs for both Model I and Model III users. Model III users have to run the Convert utility before they can make a working copy of Newbasic. The disk also includes four sample programs written in Newbasic that display its capabilities.

The distribution disk does not contain a Newbasic program. It has two programs, Creator I and Creator III, each of which builds a Newbasic program for the appropriate computer. When you run the Creator program, it presents you with each Newbasic command, a brief description of that command, and the opportunity to include or discard that command for the Newbasic disk you're making.

Once you make all the choices, the program tells you the total size of the Newbasic program you've made. You must then specify a file name under which to save the Newbasic program.

Newbasic executes exactly as Disk Basic does. You must boot the system, answer the options required by the operating system you're using, then specify Newbasic instead of Basic. From here on, Newbasic operates identically with Disk Basic, but with the inclusion of the commands you have chosen.

Program Description

Newbasic's commands fall into three general areas: graphics commands, program development aids, and command enhancements.

Fully half of Newbasic's commands are for graphics and sound generation. Additionally, several of the enhancements and program development aids lend themselves to rapid and easy handling of screen graphics. The graphics handling abilities of Newbasic are quite impressive.

You can draw circles, ellipses, and arcs by using a single command. You can even construct figures larger than the video display, although you can display only a portion of such a figure at any one time.

The command you use is Circle. You can modify Circle with up to seven parameters, specifying the center point, radius, rotation angle, and, in the case of ellipses, the radius along each of the X and Y coordinates.

The Draw command draws straight lines. You can state 15 different parameters to modify this command. These parameters let you draw lines vertically, horizontally, or at angles that bisect the X,Y coordinate.

You can draw straight lines at any other angle by specifying the starting and ending points of the desired line. If you like, you can define the starting and ending points as points off the screen.

In effect, you can draw on a grid measuring 255 by 255. The upper left corner of the video screen is 0,0 and the lower right corner of the screen is 15,63. One of the nice effects of this is that you can make graphics rotate around this universe of 255 by 255, but they are in view only when the figure is in the portion represented by the screen.

Draw is not limited to producing single straight line segments. By using several parameters with the command, you can specify complicated figures with a single Draw.

The most interesting parameter for Draw is the X parameter that lets you define a figure as a string expression. For example, you can draw a simple rectangle with the command DRAW "R20,D20,L20,U20" (which means draw a line right 20 graphics blocks, down 20 graphics blocks, and so on). Alternately, you can define the rec-


WORDSTAR®
TRS-80™
THE CONNECTION HAS BEEN MADE
For years the only REAL name in microcomputer word processing. Now it's available for the first time on your TRS-80™ Model I or III; also works on the Model 4 in the Model III mode. Introductory offer is a spectacular 50% off the suggested retail price.
only $249.00
plus $5.00 Shipping & Handling
Inquire today about WordStar and other products available from Logical Systems, Inc.
* WordStar is supplied on a Runtime version of smal-LDOS.
* Prices and specifications subject to change without notice.
* DEALER INQUIRIES WELCOME
* TRS-80 is a Trademark of Tandy Corp.
* WordStar is a product of Micro-Pro Int'l. Inc.
* smal-LDOS is a product of LSI.
LOGICAL SYSTEMS INC.
251
Logical Systems, Inc.
8970 N. 55th P.O. Box 23956 Milwaukee, WI 53223 (414) 355-5454
Art by Mark Paternostro

Phoenix

an advanced personal computer

Basic $599 kit (not shown) includes:

- Software compatibility with TRS-80 Model III and Model IV, plus CP/M
- 128k RAM card (64k normal plus 64k bank-selectable), less RAM
- 80 x 24 and 64 x 16 U/L case alphanumeric displays (software selectable)
- Z-80 CPU, with bootstrap ROM and hardware/software selectable 2 MHz and 5 MHz clock
- High resolution 512 x 256 graphics circuitry, with alphagraphics (less 16k high resolution RAM)
- Disk controller for any mix of up to four disk drives (5-¼″/8″, single/double sided, single/double density, built-in/external)
- Parallel printer and light pen interfaces.
- Built in audio
- Provisions for readily available system ROM
- Tan polyurethane enameled metal enclosure, with power supply
- Standard typewriter keyboard, plus numeric keypad
- CPU board, with six expansion slots
- Parts kit, including ICs, sockets, fasteners and mounting hardware
- Assembly manual

Complete $1699 kit shown includes:

- High-resolution 12″ green screen monitor
- Two SSDD slimline 5-¼″ floppy drives and power supplies
- Hand rubbed, solid walnut end panels
- RS232 Interface board
- System ROM
- 128K system and user RAM
- 16k high resolution graphics RAM

Additional Options

- Single or dual built in slimline 5¼″ SSDD or DSDD floppy drives and power supply
- Single or dual external slimline 8″ SSDD or DSDD floppy drives, enclosure, power supply
- 128k bank selectable RAM board (for 256K total)
- 4164 RAMs
- Monitor
- Color Graphics (available 12/83)
- Light Pen (available 10/83)
- Hard disk host adapter (available 10/83)
- Factory assembled units (available soon)

PROGRESSIVE ELECTRONICS

537 East Main Street • Lancaster, Ohio 43130 • Tel: (614) 687-1019

TRS-80 is a Tandy trademark   CP/M is a Digital Research trademark   Dealer Inquiries Invited

✓143

tangle as a string: BOX$="R20, D20,L20,U20". Then you can reproduce the rectangle anywhere in the program with the command DRAW "XBOX$".

You can do a little more magic by specifying scale, which can cause the box to be larger or smaller by factors of ¼, with a maximum range from 1 to 255. In other words, the normal size rectangle (scale defaults to 4) could be up to 63 times as large by setting the scale to 255 with the command DRAW "S255,XBOX$".

Over 20 other graphics commands exist, including GSAVE to save a graphics memory into a disk file and GLOAD to place the graphic file back into memory. PGET saves a specific graphic into a memory array, PPUT restores the array back into the video memory, and Paint fills in a specified video area, making it all white or all black.

These and the other graphics commands make possible the fastest graphics you can get in a Basic program. Combine the fast graphics animation with the five-octave range (from D below middle C to E three octaves above middle C) of the sound commands and you can achieve truly spectacular games effects.

The program development aids in Newbasic include 12 new commands. One of these, QUICKEY, enables single-key entry of 39 different Basic commands. For example, when you use QUICKEY, holding the clear key while pressing the 4 key causes CHR$(to appear immediately on the screen.

Holding the clear key while pressing the 5 key displays INKEY$, and pressing C displays CMD". It takes a while to become accustomed to QUICKEY, but once you gain familiarity with it, you'll find it a real time saver.

Another nice feature, the DEFKEY, lets you use any 10 keys to call an entire string of characters. One key can call a maximum string length of 127 characters. All 10 keys can call a total of 221 characters.

NTRON is a program tracing routine that lets you set a specific range of program lines you want traced. Additionally, you can specify any expression that you wish traced, and NTRON returns the value of that expression each time the program encounters it.

Yet another tracing function is SPOOLON used in conjunction with the PON command. PON sends everything displayed on the screen to the printer. Preceding PON with SPOOLON sends everything from the screen to a disk file. You can then redisplay the disk file on the screen or send it to a printer. This way, you can make a permanent copy of the working of a program.

The handiest program development command is LOC.. LOC. followed with a string you want found searches a Basic program for that string. For example, if you want to find and change X$ to Z$ in a program, you could find X$ with the command LOC."X$".

Newbasic searches for the first occurrence of X$ in the program with which you are working. You can change X$ to Z$, then look for the next occurrence of X$ with the command LOC.. If you want, you can do this for the entire program.

Among the Basic enhancements are the commands Call, DPEEK, DPOKE, and Do...Until. You use Call to call a machine-language subroutine. With this command it isn't necessary to use DEFUSR to identify the subroutine. You can locate the routine in high memory at will.

You can follow Call with the exact memory location of the routine in either hexadecimal (hex) or decimal form. Alternately, you can follow Call with an expression that corresponds to the routine's location. You can use any number of routines with this command.

Think of DPEEK and DPOKE as double PEEK and POKE. These commands return and insert 2 bytes into memory at the specified location and at that location plus one. The number returned is in the most significant byte, least significant byte form.

Do...Until lets you loop a routine that continues as long as the test following Until isn't met. The Until test can be any logic or arithmetic expression that you want tested. You can nest up to 10 of these loops (one Do...Until expression inside another).

The manual for Newbasic is a spiral-bound book showing each command. It includes an explanation of the command and one or more specific examples of its use.

The explanations are reasonably simple, but you'll need some knowledge of Basic and of programming to understand them. This isn't a serious flaw, as few neophyte programmers have any use for a set of enhancements like Newbasic.

The manual's best feature is its installation instructions. These step-by-step instructions tell you how to install the program, with specific instructions for TRSDOS, LDOS, DOSPLUS, and NEWDOS80.

Conclusions

Newbasic seems to be a reliable program. Try as I might, I couldn't get it to crash. Nor was I successful in finding any bugs in the time I was able to use Newbasic.

The program development tools I discussed are fairly valuable. The trace functions could have saved me hours of debugging time if I'd had them in the past.

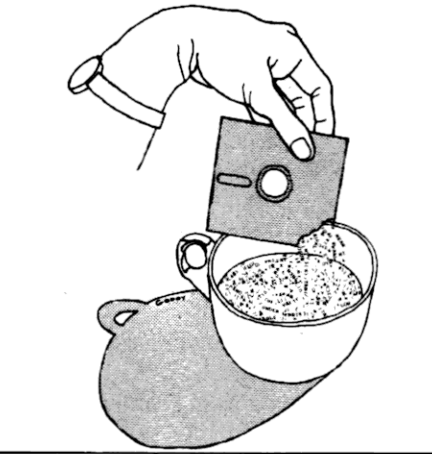
The spooler functions are much easier to use with a Basic program than the spooler that comes with NEWDOS80. And once you've become familiar with the use of QUICKEY, it saves hours of typing when you are keying in a Basic program.

If you're writing a program that requires any amount of screen graphics, Newbasic greatly simplifies your task. However, the graphics commands take some practice to use easily. Most have several optional parameters, and the results of the parameters can be surprising.

Circle is initially confusing. It begins drawing each circle from the three o'clock position on the screen. For the last 23 years, in flying and in reading blueprints, I've used the 12 o'clock position (top of the screen) as zero degrees for circles and arcs. It took quite a while for me to consistently get the results I expected from the Circle command.

Once you become accustomed to the commands, they are fairly easy to use. You'll probably need to keep the manual handy if you're attempting complicated graphics, but straight lines are a snap. With a little practice, you'll find that setting up formatted screens for user input is actually easy with Newbasic.

The program's greatest value, however, is as a game development tool. The rapid graphics and sound generator make near-arcade results easily achieved. ■



★★★★

Instant Assembler
Mumford Micro Systems
P.O. Box 400
Summerland, CA 93067
Models I and III
32K RAM
$39.95 cassette
$49.95 disk

by William J. Schauert

Instant Assembler, a complete Assembly-language development system, provides many enhancements to the Radio Shack Editor/Assembler. It's a good system for beginning Assembly-language programmers, and an adequate system for advanced programmers.

Instant Assembler is unique from other editor/assemblers in that it assembles code as you type it in, line by line. This way, you catch mistakes without the tedium involved in re-editing to correct errors that occur on assembly and later debugging the program for logic errors.

Instant Assembler contains an editor to create your source files, a built-in symbolic debugger, and a linking program that lets you create separate modules and link them together into a single program.

I tested the program under TRSDOS 1.3 on the Model III, and under LDOS 5.1.3 on my MAX-80. The manual states that the program runs under most popular DOSes for the Model III.

This package includes a program disk and a 65-page user's manual. The program disk is a TRSDOS data disk. For those of you with a single drive system, the disk self boots into a special program that transfers the programs to your system disk.

The programs include the Instant Assembler program, three versions of the linking loader, a stand alone version of the debugger program, and a program to restart the assembler with the source file in place (if you have to reset during a debugging session). The main program, DSKIAS, contains the editor/assembler and a built-in debugger.

In the traditional method of creating an Assembly-language program, you enter your source code with an editor, then run the assembler. The assembler checks the syntax of each source statement, and generates machine code or an error message. You then re-edit the source file to eliminate errors and try the assembler again.

Once you have an error-free source file, the assembler generates object code, the machine code for execution. Since neither the editor nor the assembler checks for logic errors, you must debug the object code. Most operating systems include a debug program.

The author of the Instant Assembler takes a different approach to Assembly-language programming. This program assembles the source line when you hit the enter key while in the edit mode.

It checks for proper Z80 syntax of the opcode and operand data, and it looks for any duplicate use of the label. It also checks the range of relative jumps. If you have any errors in the line, the system reports the error and places you in the line edit mode with the cursor at the first character in the offending field.

After you finish your source file, you can assemble it directly into memory and debug your program with the built-in debugger. Since the source file is still in memory, you can also debug using the symbol names in your program.

If you discover an error, you can return to the editor mode, correct the bug, and try again. Having all these programs in memory at one time helps speed up the process of developing an Assembly-language program.

The Assembler

The assembler portion of the program is actually both editor and assembler. The editor accepts text from the keyboard and places it in the proper fields for the assembler. The input data is in the form of Z80 mnemonics and comments.

✓ See List of Advertisers on Page 291

The manual covers each of the 37 editor commands in enough detail to let even a beginner operate the program. If you're familiar with the TRSDOS Basic line editor, you already know how to use most of the line editing commands.

The assembler expects the input line to follow a fixed format. You can enter an optional six-character label as the first field. If the first character of the label is the ampersand, other program modules can use this label. In Z80 talk, this label is global or public. If the label doesn't have an ampersand, it's known only to the current program module.

The next field is a four-character opcode field that can contain any known Z80 opcode. It can also contain two undocumented Z80 opcodes that let you access the upper and lower halves of the index registers separately.

The last field, the operand field, can contain up to 45 characters and include any type of operand data or comments. If you want a comment, enter a semicolon followed by your comment. However, you don't need to align your comments; the editor does this automatically.

The editor/assembler works quickly and efficiently to help you produce Assembly-language source code. On a global level, the editor has commands to insert code, delete lines, and move blocks of code from one area to another.

It's possible to list your source code to the display or the line printer. When you send code to the printer, the program paginates it, but doesn't add a title to your list. You can also save code to the disk or tape. You can save source code in the Instant Assembler compressed format (a semi-assembled code) or in standard Radio Shack EDTASM format.

If you choose the latter, the program saves the source in ASCII format with line numbers. You can then edit this code with Radio Shack's EDTASM.

You can also include other sections of source code in your current program by using the Merge command, which appends one Instant Assembler file on disk to one in memory. As the program adds the source code, it checks for Z80 syntax errors, just as if you entered the code from the keyboard.

Other assembler commands let you read the disk directory, delete a file from the disk, and find instructions in the code that reference a certain label. Another command lets you assemble your code directly to memory so you can debug it as soon as you finish without saving and then reloading it with a debugger program. You can also save your object code to disk or tape when you have a finished product.

The assembler allows only a few pseudo-ops, including the Z80 types of DEFM for messages, DEFB for byte data, DEFW for word data, DEFS for storage allocation, and EQU for symbolic equates. These have some surprising limitations.

You cannot use the EQU pseudo-op to define byte data. A statement like:

```
LINEFEED EQU 10
LD A, LINEFEED
```

is not allowed in this assembler. You can assign only word length (16-bit) equates.

You can postfix symbols with an expression, but the range is limited to − 31 to + 287. This means that LD HL,(BEGIN + 31) is allowed, but an expression like LD HL,(THEEND BEGIN) isn't allowed.

The storage allocation is limited to 4,095 bytes in a single DEFS statement, but you can use multiple statements to obtain greater allocation areas. The last limitation is that a DEFM statement can only define a string of 43 characters due to the length of the operand field.

You can break your message string up into several statements of 43 or fewer characters. The assembler also does not recognize any ORG, End, or DEFL pseudo-ops. The assembler automatically supplies ORG and End when you list the program.

The Debugger

After you assemble your program into memory, you can switch to the built-in debugger program called MicroMind. This program is also furnished in a stand-alone version. MicroMind has over 21 commands. As with the assembler, the manual explains each in detail.

The main routine used in the debugging process is the Step command. This mode lets you step through your program in half steps, each step broken up into a Fetch and then an Execute cycle. You can step through any area of the program including sections that might be in ROM.

When you enter the step mode, the program asks you for the first address. After you enter a valid address, the program fetches the Z80 opcode at that address. The CRT displays the machine instruction, the disassembled code, and the contents of all registers and most flags.

When you hit the enter key, the program executes the instruction. The display keeps the old information and shows the new register information. This lets you compare the before and after effects of the instruction as the program executes it. Some commands let you fast step up to 99 steps, and execute through a call that has already been tested.

MicroMind also supports traditional breakpoint debugging. This lets you set a break location in your code, then run the program until you reach the break location. The display then shows the current state of the registers. The breakpoint is not restored when you reach it; restoring a breakpoint requires a separate command.

Two blocks of memory appear on the display at all times. These blocks are 19 bytes long and you can set them to start anywhere in memory. This is handy if you want to keep an eye on a buffer in which you're changing data.

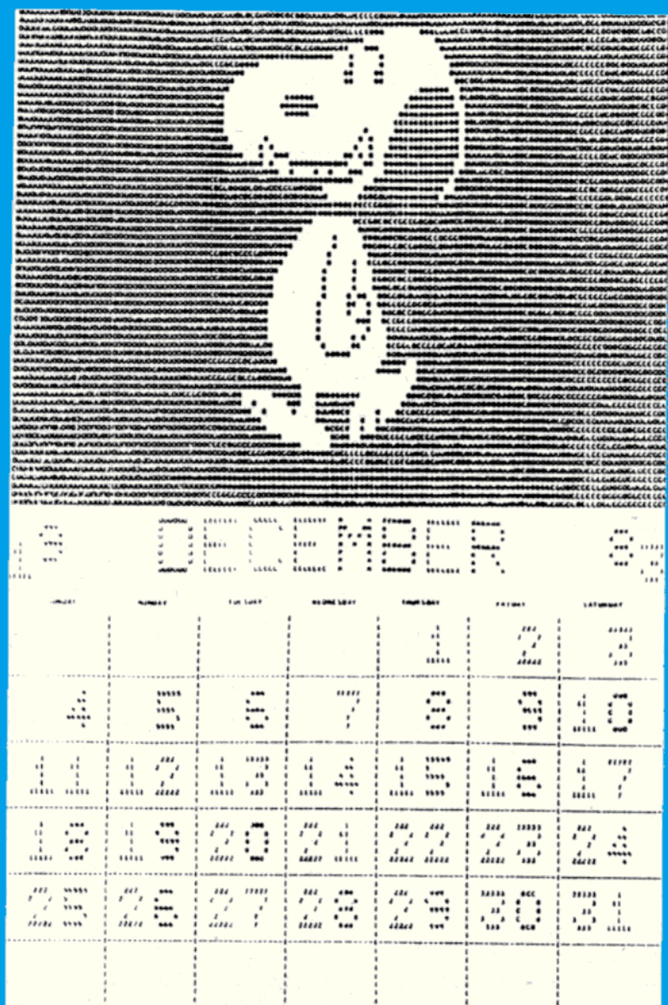
MicroMind also lets you display a block of memory in ASCII so you can detect string information. The display is 50 characters long (five lines of 10 each), and you can move it forward or backward with the up- and down-arrow keys. Other commands let you modify memory and the processor registers.

Instant Assembler includes some utility routines in the debugger program. The Find Number routine lets you search memory for any 1- or 2-byte sequence.

The Disassemble command shows you the Z80 mnemonic code starting at any first address and incrementing by one instruction each time you press the enter key. The program sends this information to the screen or to your printer. The debugger also has Hex-to-Decimal and Decimal-to-Hex number conversion routines.

The MicroMind program is easy to use and a good debugging tool. It's much better than Radio Shack T-Bug and good competition for the disk-based debug program included with TRSDOS. The disassembler feature is

CARTOON CALENDER



☆ HOLIDAY SPECIAL ☆

- Just enter the year desired and this program will print a complete calendar, with a beautiful graphic picture of your favorite cartoon characters for every month.
- Print calendars for fun or profit for yourself, and for your friends and family.
- Makes a great holiday present!
- Hanging a calendar in your home or office will make a great conversation piece. People will be sure to stop and say "Hey did you make that!"
- Source code included – versions available for the Model I, II, III and IV, tape or diskette, on 80 and 133 column printers, and 80 column printers with 133 character compress mode.

SEND FOR IT TODAY!
– ONLY – **$30.00**

(Tax and shipping included.)

CHECK OFF YOUR TYPE OF SYSTEM ►

NAME: ______________________
ADDRESS: ______________________
____
____

	TAPE	DISK
MODEL I	☐	☐
MODEL II	☐	☐
MODEL III	☐	☐
MODEL IV	☐	☐

Internal Storage (______K) ?
80 Column Printer with 133 Character Mode ☐
Standard 133 Column Printer ☐

Make check or money order payable to:
BLUE PRINT SOFTWARE
100 Wauwepex Tr.
Ridge, N.Y. 11961

482

nice because you can see your code as you write it in source as well as in machine code.

The only objectionable item in the debugger is that the Z80 prime registers aren't included in any of the displays. To view these, you must execute an exchange opcode and display the registers again.

The Linking Loader

An old computer saying states: "A program expands to fill available memory." Well, in case it does, the Instant Assembler package has a program to help you manage large programs.

With the Linking Loader, you can break your source files up into related sections, define entry labels by prefixing the ampersand to them, and combine the entire program at the end to create your final object code.

The Instant Assembler contains several versions of the linker to create programs that start at the top of memory and work their way down, or start in low RAM and work toward the top of memory. All the linkers use the Instant Assembler source as their input file. You must save your modules in compressed format to use this feature.

As you load each module, the program checks it again for proper Z80 syntax, for ranges of relative jumps, and for duplicate labels. At the end of the loading session, you receive a summary of error count and a list of undefined labels.

You can also search for the location of any symbol with the Symbol command. This lets you find the absolute address of any global symbol. If you like, the linker sends a list of all global symbols and their addresses to the printer. After you have all your source modules linked, you can save the resultant object code on tape or disk.

Conclusion

The unique feature of this assembler is the instant assembly of your source code as you enter it. This is a good feature for a new Assembly-language programmer, because it immediately eliminates simple syntax errors from your source code.

The editor lets you save your source code in the Instant Assembler compressed format or in Radio Shack EDTASM format. If you choose to save your programs in compressed format, you can save about two and a half times the number of bytes of the EDTASM format on your tape or disk.

The only serious problems with the assembler are the limitations on the range of expressions and the restriction to 16-bit equates. These problems become severe for commercial applications, but might not affect the average home user or beginner.

The MicroMind debugger is easy to use and rivals almost any other debugger available for the Model III. The source code linker lets you break up the program into smaller sections, and later combines them into a single object program for execution.

The documentation includes a complete description of each command and examples of how to use each one. The manual uses a sample program as an example of all other portions of the development system, including the debugger and the Linking Loader.

Because of some compatibility limitations between this assembler and the Radio Shack EDTASM assembler, the manual devotes a chapter to adapting your programs to EDTASM. It would be more helpful if the author of the program corrected these limitations rather than discussing how to program around them.

In general, the Instant Assembler is a good Assembly-language development system. It provides many enhancements to the Radio Shack EDTASM system.

It also works quickly and improves user efficiency by having the editor/assembler and debugger in memory along with your program. This is a good starter system for beginning Assembly-language programmers, and a usable system for advanced programmers. ■

★★½

Model 100 Games #1
SilverWare
P.O. Box 21101
Santa Barbara, CA 93121
24K Model 100
$24.95 cassette

by Eric Grevstad
***80 Micro* staff**

If you've been waiting for high-quality, fast, machine-language games for your Model 100, keep waiting. In the meantime, SilverWare (a spinoff of *CLOAD* and *Chromasette* magazines) has converted a quartet of simple Basic programs for the portable—one or two of which are modestly entertaining if you have modest expectations.

The cassette includes two copies each of two adventures and two graphics-oriented games. All loaded easily on the first try; the documentation includes a handy table showing their locations on a Radio Shack CCR-81 recorder's tape counter, as well as the memory requirements for each.

The games, Blockade and Reversi, can share RAM with other programs on the menu, but each of the adventures, Alexis and Frankenstein, fills most of a 24K machine.

By that criterion, Alexis (18,500 bytes) wins the Not Worth Killing Everything But ADRS.DO Award. The plot is interesting: As the imprisoned son of the late king, you must escape the usurping General Tarkaan, sail to each of four neighboring islands to collect an army, and then return to battle Tarkaan, surviving in effect four adventures and a combat strategy game.

But Alexis combines this sophisticated premise with the clumsy syntax and limited range of an antique (1980, perhaps) Basic adventure. Given the 100's 40-column screen, even terse two-word commands are too long to fit on the same line as the windy "What is your command, Alexis?" prompt.

Other nagging lines—"You can't go that way, Alexis"; "You have boarded the boat, Alexis"—make the name begin to sound like fingernails on a blackboard.

Alexis has annoying random elements, too: Sometimes Pluto gives you the Crystal Sphere of Hades if you bring him the giant pearl, but sometimes he's cranky and demands something else.

While sailing from island to island, you have to keep an eye on the weather and type WAIT if it looks stormy. Even if you do, the program occasionally kills you off, saying "You should have waited."

Adventurers used to brain-busting

Quality you expect, at a price you don't.

BECK DOUBLE DENSITY DISKETTES

SINGLE SIDED **$2.19 ea.** / **$2.79 ea.** DOUBLE SIDED

Our message to you is simple. If you like the quality of Dysan, Verbatim, 3M, et al, you'll like the quality of Beck soft sector, 5¼″ flexible diskettes. The only major difference is cost. We're less expensive. In fact, a lot less expensive.

Why does Beck cost less?

Our philosophy is: Excellent quality and reliability, at a cost that beats the jackets off other diskettes. We can do it because we (1) put our money into the product, not mega-marketing schemes and fancy packaging; and (2) sell our money-saving 25-diskette pack to you direct via a toll free order line, so you get fast, door-to-door service efficiently.

When you buy Beck, you've got the best.
Beck Quality. Beck Reliability.
And, of course, Beck Price.

1D, soft sector 5¼″ diskette $2.19 each
2D, soft sector 5¼″ diskette $2.79 each

For IBM, Apple, TRS and 97% of popular microcomputers.

What about quality and reliability?

At Beck, our success as a diskette manufacturer depends upon our ability to provide you with a fully reliable, quality diskette – every time. For that reason we take no shortcuts. You get the best because we are committed to excellence. Every diskette is manufactured to very strict quality standards. We test and retest 21 times throughout the manufacturing process to insure compliance with no less than 42 rigid specifications. We make sure you get the very best – a 100% certified, 100% error free diskette.

Our satisfaction money-back guarantee and full 7 year warranty are proof of our commitment to excellence and confidence in our product.

(in New Hampshire call 924-3821) ***Door to Door in 48 hrs.***

Order Now Toll Free

VISA MasterCard COD'S CASH ONLY

Corporate Accounts Welcome

1-800-BECK-MFG

Order Toll Free 1-800-232-5634. Available in 25 pack only, plus freight. Complete with hub reinforcing rings, Tyvek envelopes, color coded user lables, and nonmetallic write protect tabs. All Beck Diskettes meet or exceed ANSI specifications.

puzzles will be sorely disappointed. Alexis begins his odyssey in a jail cell, where the guards put him without removing the skeleton key he carries.

On the first island, I worried about natives or monsters but found only stuttering strings: "Alexis, you are on the shores of the island of Skiros on the island of Skiros."

By contrast, while no one would call Frankenstein a great adventure—simple two-word commands ("Unlight candle"), slow Basic that lets you type in a command before the display's processed your previous one—it's a pleasant few hours' exercise.

Frankenstein (15,000 bytes) is relatively easy, but has a couple of clues or items that form a good introduction to adventuring logic. I can't recommend it for children, though, because it involves digging up and hacking at corpses. As the last surviving relative of Victor Frankenstein, you must finish his work and activate the monster, who's short a few parts.

But once you get into it—scurrying between the mansion, the graveyard, and the old mill; trying to get at and open the safe; and dodging quicksand, werewolves, and finally the awakened monster—Frankenstein is ghoulish fun.

Like Alexis, however, it might tempt portable players to cheat. For people used to Model I/III disk adventuring, it's all too easy to hop into Basic for a simple LIST. (That's how I learned the trick of getting past the wolf; I'd never have been gruesome enough to think of it myself.)

Both Frankenstein and Alexis have one wonderful feature, the best thing about Model 100 adventuring—you can save a game in progress to RAM (your position is stored as a bunch of numbers in a 350-byte do-file), replacing disk or cassette I/O with "Save game" or "Load game" and a tap of the enter key.

Key work isn't so easy in Blockade (6,500 bytes), a semi-Centipede game in which you steer a snake around the screen, trying to reach targets and growing longer as you do so. The targets appear and disappear randomly, sometimes for no longer than an LCD flicker; each is worth a random one to nine points.

Hit the wall or your tail, or your opponent in two-person games, and you lose five points. Blockade then displays your current score and restarts with new snakes.

The word that springs to mind, especially after the opening instructions take eight seconds to appear, is "slow"; only the fastest of three speeds is at all interesting, and sluggish response makes maneuvering difficult. Rather than the Model 100's arrow keys, the Esc, tab, /, and shift keys steer single players up, down, left, and right respectively.

Two-player games are awkward. Not only is the 100's screen hard for two people to see at once, but the second player has to use the apostrophe, slash, comma, and period keys to steer, reversing the order (left hand up/down, right hand left/right) of the other arrangements or the Model III arrow keys.

As if that weren't enough, the game drags on until someone reaches 100 or −100 points, or until a player leads his opponent by 100 points. This can take some time. The playing options and default values (one player, sound, medium speed) are nice touches, but Blockade remains basically a dull game.

Reversi, on the other hand, is the same old Othello that's been on micros for years, but the Model 100 version is a first-rate use of 5,250 bytes. The documentation is poor—I'd never played Reversi before and learned by trial and error—but the game is neat, simple, and fiendishly challenging.

On a Model I or III, Reversi was interesting. Scaled to fit in your lap, it's an ideal armchair or plane-ride diversion.

While two can play, Reversi is best as a solitaire duel against the 100. A # sign, steered around an 8-by-8 grid with the arrow keys, lets you enter your move, capturing enemy pieces between two of yours. Then the 100 takes its turn, moving the # sign around the board as if considering moves before making its choice and capturing even more pieces.

Games become grim, vicious fights for the crucial edge and corner spaces; toward the end, the lead swings wildly back and forth until a musical salute announces the victor. I've managed several wins and a tie, but I'm by no means through playing. Of the four SilverWare games, Reversi is the only one I'm keeping in my Model 100.

I can recommend Reversi because it takes new life in a lap-sized version, but the other Model 100 games have no real attraction except portability. Probably soon there'll be adventures with plots as elegant as the save-to-RAM feature, or arcade games that aren't anchored by plodding Basic. The Model 100 will be a great game computer. It just isn't one quite yet. ■

★★★★

Microspooler Buffer/Interface
Consolink Corporation
1840 Industrial Circle
Longmont, CO 80501
$199 (16K parallel to parallel)

by Mare-Anne Jarvela
***80 Micro* Technical Editor**

If you've ever waited impatiently for your printer to stop printing so you could use your computer again for other tasks, here's a microspooler that makes work much easier for you.

The microspooler stores data until the printer is done, letting you continue your work. Data acceptance rate is up to 5,800 characters per second.

You can also use the microspooler between a phone modem and a printer, or the modem and your computer.

The four spooler models available include parallel to parallel, parallel to serial, serial to serial, and serial to parallel configurations. The model I describe in this review is parallel to parallel.

You can configure each model to store 16K, 32K, or 64K of data. The price is slightly higher for 32K and 64K, and serial models are more expensive than parallel.

The microspooler is small: 6.3 inches high by 2.6 inches wide by 8.3 inches deep. It's a white box with black edges that weighs two pounds, 15 ounces and stands vertically to save space.

A two-digit numeric display on the front is the status readout that tells you how much data is in the spooler or how many copies are left to run. On the front you'll also find the copy/pause button and the reset button.

"The microspooler stores data until the printer is done, letting you continue your work."

The back has two ports, one for input and one for output. The on/off switch is also on the back. Connectors for parallel ports are 36-pin Centronics compatible; the input port is the receptacle and the output port is the plug. For serial ports, the input port is the DB-25S receptacle and the output port is the DB-25P plug.

Using the Microspooler

The microspooler hooks up to most standard printers, but lack of standardization in the printer industry might give you problems in getting the right kind of cable for your printer. Consolink is helpful if you need advice or help in building your own cable.

It's simple to hook up a parallel printer and microspooler. With the serial microspooler or printer, you have to set a few switches. The serial ports also have selectable baud rates. The manual is easy to follow and tells you exactly what to do.

When you first turn on the microspooler, you can perform a self-test. Press and release the reset button while holding in the copy/pause button. This is a software test of ROM (read-only memory).

If this test is successful, your printer tells you ROM is OK and how much memory is available. Now you can start filling your buffer.

The minute you start sending data from your computer to the printer, the microspooler fills up and control returns to the computer. The printer starts printing simultaneously. The status readout changes from 00 to 01 (1K) and so on as you send data.

When you stop filling the buffer, the readout decreases until it reaches 0 when the printer finishes printing. Reset the buffer to 00 if you don't want an extra copy. While the printer is running, you can press the copy/pause button if you need to change the paper or make adjustments.

When printing is complete, press the copy/pause button for one more copy of whatever is in the microspooler's memory. It's also possible to preset the number of copies (up to 99) you want before you start printing. Press the copy/pause button until the desired amount of copies appears on the readout.

You also have the option of pausing after every copy. The microspooler also has a pause on form feed function.

I found the microspooler simple to install and easy to use. It's a helpful tool if you have a lot of printing to do. The microspooler frees you and your computer to do other things while the printer works. All in all, the microspooler is a good buy. ■

THE WAIT IS OVER

CP/M®

EXCLUSIVELY FOR THE

TRS-80®

Model 4

Now, for the first time, unleash the powerful features resident in your Model 4 computer. Open up the vast store of CP/M software such as WordStar®, dBASE II and Multiplan™, along with thousands of others.

- Includes INTERCHANGE ™, a utility that allows reading, writing and copying 20 different manufacturers' disk formats such as IBM, KAYPRO, OSBORNE, XEROX, etc.
- Includes MEMLINK™, a unique feature that uses the optional 64K RAM memory as a fast disk drive.
- Complete with all these CP/M utilities; ASM, DDT, DUMP, ED, LOAD, PIP, STAT and SYSGEN.
- Operates at the 4Mhz clock in the standard Model 4 mode.
- NO HARDWARE MODIFICATIONS. Just insert the disk and boot.
- NO COPY PROTECTION. Backups may be made for your own use and protection.
- The CONFIGURATION program supports a full range of 5-1/4" disk drives: 35, 40, 77 and 80 tracks, single and dual sided in any combination as well as the standard Model 4 drives.
- Includes MODEM 7, a powerful public domain communications program for file transfer and remote data base access such as CompuServ and the Source.
- Supports 80 x 24 video, reverse video, direct cursor addressing and more.
- Utilizes the Model 4 function keys and allows user defined keys.
- Auto Execute command for turnkey applications.
- FORMAT utility permits up to 52 disk formats to be constructed, all menu driven.
- Fast backup routine with verify for mirror image copies.
- All support programs are menu driven for ease of use.
- Ready to run in the standard 64K Model 4. The additional, extra cost, 64K RAM upgrade not required.
- Complete with over 250 pages of comprehensive user documentation.

AVAILABLE NOW FOR IMMEDIATE SHIPMENT.......................................$ 199.95

The full line of MicroPro software is now available formatted for the Model 4 using our CP/M. Each disk is already configured and ready to run. Just install the printer of your choice and go.

WordStar® Fast memory mapped version	$250	InfoStar™ Advanced DBMS	$250
MailMerge® Multi-purpose file merging program	125	ReportStar™ Report generator & file manipulator	175
SpellStar® 20,000 word proofreader on a disk	125	DataStar™ Data entry and retrieval package	150
StarIndex™ Creates index and table of contents	95	SuperSort® Fast and flexible sorting is yours	125
WordStar Professional. All the above for only	450	CalcStar™ Advanced electronic spreadsheet	95

ORDER INFORMATION

Call now and your order will be shipped at once from our Dallas warehouse. We accept American Express, MasterCard, Visa, and most any other form of payment known to man. Credit cards are not charged until your order is shipped. Add $4 UPS surface shipping and handling on orders within the 48 States. No State Sales Tax on software or shipments delivered outside of Texas. No refunds. Defective items are replaced upon return, postpaid.

ORDER NOW . . . TOLL FREE

800-527-0347
800-442-1310

The Toll Free lines are for orders only.
Specifications subject to change without notice.

128K MEMORY UPGRADE

Our upgrade includes 64K of 150nsec RAM, genuine PAL® chip and instructions for installation. This kit will upgrade your 64K Model 4 to 128K and allow the use of our MEMLINK and TRSDOS 6.x MEMDISK. Comes with a full 1 year guarantee.

A BARGAIN AT ONLY $99.95

AD #883

MONTEZUMA MICRO

214-339-5104
Redbird Airport, Hangar #18
P.O. Box 32027
Dallas, Texas 75232

✓411

"WE KEEP YOU RUNNING"

✓ See List of Advertisers on Page 291

★★★★½

Using Scripsit
William J. Haga
Wadsworth Electronic Publishing Co.
10 Davis Drive
Belmont, CA 94002
Softcover, 250 pp.
$21.95

by S.F. Tomajczyk
***80 Micro* staff**

It's said that you can't teach an old dog new tricks. It seemed true until I began reading *Using Scripsit* by William Haga. That's when I realized that, although I've actively used Scripsit for the past year, I haven't really been using it.

Using Scripsit is truly user-friendly. It represents what a reference book should be: comprehensive and informative concerning the subject matter, readily accessible for locating specific information, and most importantly, communicative in an understandable fashion.

The book discloses everything you ever wanted to know about using Scripsit's capabilities on your Model I or III. The information applies only to Scripsit versions 3.2 and 1.0, so if you have Scripsit on a Model II the book won't be of any use to you.

Using Scripsit employs a hands-on approach to learning to get the most out of Scripsit. You should read it with your computer in front of you. William Haga presents each procedure in numbered steps that tell you what to do, what your video display looks like when you do it, and how the printed result looks on paper.

You'll never feel lost or confused. And to make certain that you won't, Haga includes exercises to test your growing skills in using Scripsit.

At the end of each chapter, a Common Mistakes section lists the problems and solutions you're most likely to encounter. Also, the left margins are filled with comments that clarify the text, provide helpful and humorous tidbits, and direct you to other sections of the book for further information.

At a Glance

The first section of *Using Scripsit* is an introduction aimed at the first-time user. In Chapters 1–7, Haga methodically takes the reader through the fundamentals of disk use, writing and formatting the video screen, saving and retrieving Scripsit files, basic editing commands, and formatting and printing a document.

The second part of the book deals with more complex applications for the experienced user. Chapters 8–15 include valuable information and explanations of Scripsit that you'll refer to time and time again. They cover block moves, search and replace routines, creative formatting, and chain loading.

They also clarify the forbidding headers, footers, and page numbering system, as well as clearly explain the Hot Zone of Scripsit's hyphenation function, and how to save a half-ruined printout without reprinting the entire document.

The best is yet to come: In the remaining three chapters, Haga shows you how to merge VisiCalc with Scripsit, create personalized form letters, use Scripsit to write computer programs in Basic, and print documents in special formats.

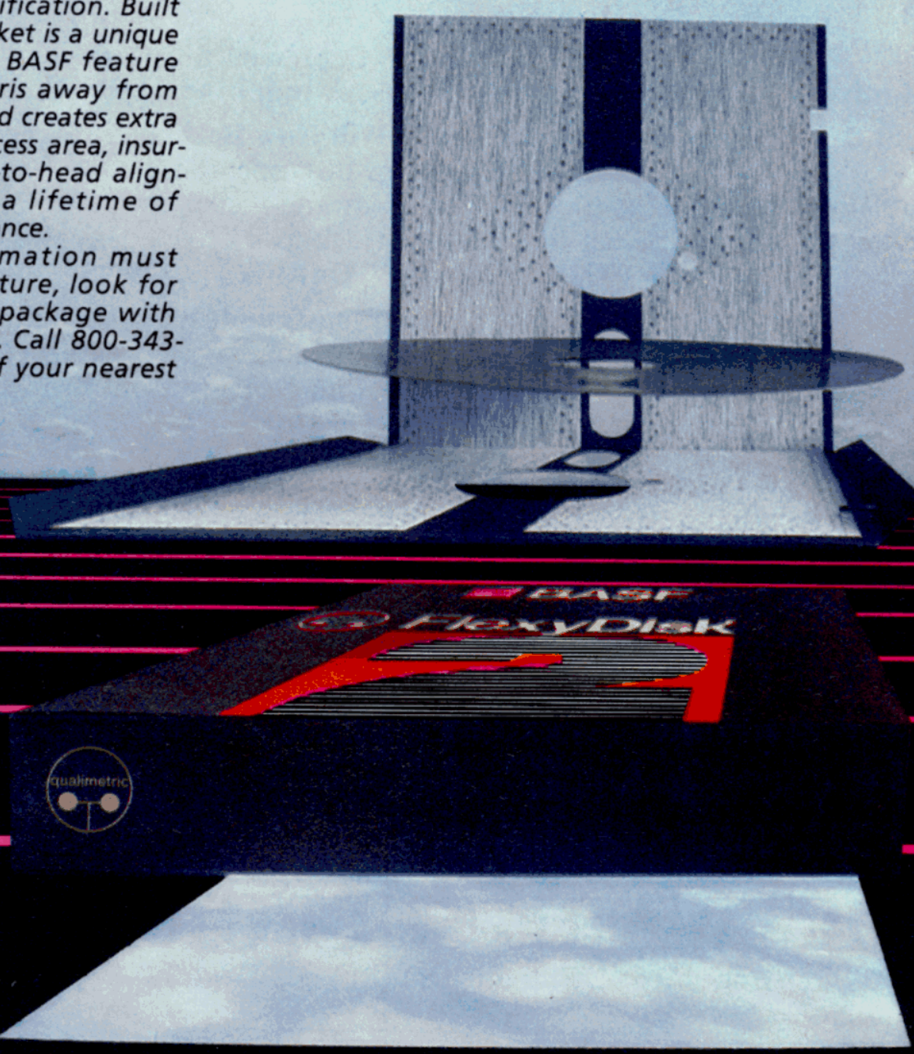
Continued on p. 61

BASF QUALIMETRIC™ FLEXYDISKS®
BUILT FOR ETERNITY - WARRANTED FOR A LIFETIME.

BASF Qualimetric FlexyDisks® offer you more...an extraordinary new lifetime warranty. The BASF Qualimetric standard is a dramatic new international standard of quality in magnetic media...insurance that your most vital information will be secure for tomorrow when you enter it on BASF FlexyDisks today.*

We can offer this warranty with complete confidence because the Qualimetric standard reflects a continuing BASF commitment to perfection...a process which begins with materials selection and inspection, and continues through coating, polishing, lubricating, testing, and 100% error-free certification. Built into our FlexyDisk jacket is a unique two-piece liner. This BASF feature traps damaging debris away from the media surface, and creates extra space in the head access area, insuring optimum media-to-head alignment. The result is a lifetime of outstanding performance.

When your information must be secure for the future, look for the distinctive BASF package with the Qualimetric seal. Call 800-343-4600 for the name of your nearest supplier.



ENTER TOMORROW ON BASF TODAY

BASF

DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES DISK DRIVES

PRICE BREAKTHROUGH

New! **Diskette "Give-A-Way?"** New!

Judge for yourself!

10 pack in Library Case — **New Low Price**

SS/DD — with hub ring

Introducing the NEW Tabor 3¼" Disk Drive

Storage 737,280 Bytes — 2 in one Case . . . . **$ Call**

SOFTWARE SUPPORT INC. presents Software

Ldos[C] Improved — New efficient improvements for your Model I/III/IV LDOS[C] System

With Our Program Enhancements You Will Now Be Able To Accomplish:

1. Double the speed of disk operations
2. Reliable fast clock boot (with enhanced hardware — such as the Holmes Sprinter™)
3. Improved dual sided and 80 track disk drive operations.

Model I/III/IV Instapatch™ On Disk $29.95

Esylst™ — Neatly and automatically print out any number of your basic programs. Page numbering, top of page, skip, no limit to the number of programs printed at one setting. Fast — Easy — Efficient. Use with Newdos[A], Dosplus[B], Ldos[C]

For Your Model I/III/IV On Disk $19.95

DRIVE A HARD BARGAIN™

We have rewritten the book on Hard Drives. Our hard drives are the fastest, the most versatile (Newdos[A], Dosplus[B], and Ldos[C] drivers), the most adaptable, (runs on the Model I, Model III and Model IV using the same adapter), but, we must admit, it is the lowest (in cost that is) starting at for a 5 meg. (formatted) primary and **$695.00** for a secondary system. All systems come complete with power supply, case, cables and Instapatch™ software drivers for fast-trouble-free installation and operation. **Nothing else to buy!**

Self Test — Automatic Diagnostics *Error Correction — ECC* *Daisy Chain up to 40 Megabytes*

Warranty — ONE FULL YEAR. Call for details

TOLL FREE ORDERING
1-800-343-8841

GENERAL AND TECHNICAL
1-617-872-9090

SOFTWARE SUPPORT, INC.

One Edgell Road, Framingham, MA 01701 (617) 872-9090

Hours: Mon. thru Fri. 9:30 am to 5:30 (E.S.T.) Sa . 10 am to 4:30 pm

TERMS:
M.C./Visa/Amex and personal checks accepted at no extra charge.
C.O.D., please add $3.00.
Shipping: Please call for amount.
Not responsible for typographical errors.

Canada
MICRO R.G.S. INC.
751, CARRE VICTORIA, SUITE 403
MONTREAL, QUEBEC, CANADA, H2Y 2J3
Regular **Tel. (514) 845-1534**
Canadian Toll Free 800-361-5155

Dealer inquiries invited.

A ™ Apparat Inc.
B ™ Microsystems Software Inc.
C ™ Logical Systems Inc.
© Copyright 1983

✓ See List of Advertisers on Page 291

Continued from p. 56

He explains, step by step, how to send Scripsit files by phone across town or across the globe, and how to manipulate your operating system's password protection system to protect your Scripsit files from prying eyes.

When I reached the end of Chapter 18, I was convinced that William Haga had a sadistic streak in him. Why else would he continue to throw so much valuable information at me in his appendixes?

One appendix provides the symptom, cause, and solution to errors you frequently encounter. It covers everything from Scripsit error messages to printer difficulties, and from drive unit problems to DOS troubles.

Another appendix lists reference sources that include computer magazines, books and manuals, and independent sources of equipment and software. And yet another appendix functions as a glossary for Scripsit users! The author had done the impossible: I actually learned some new tricks.

What exactly did I learn? Well, did you know that Scripsit has a widow suppressor? That you can change the screen width to up to 132 characters? That with the Search command you can find several variations of a word for spelling errors? That Scripsit can hyphenate wrapped-around words with its hyphenation block?

That you can create a paragraph pantry to easily and quickly customize form letters? Or that you can chain files together in order to print and number an entire document? I didn't until *Using Scripsit* told me. Now *80 Micro*'s technical editors ask me for advice on Scripsit.

With the information in this 250-page book, I've found Scripsit to be much more powerful than I thought. Now I use it to finish office paperwork more quickly, write and edit chapters in my book, create and print form letters and memos, and write Basic computer programs.

For the skeptics out there, *Using Scripsit* does have its drawbacks, but they are trivial. For instance, I feel that the chapters concerned with screen editing should have been introduced earlier in the book.

This would let you go from booting up Scripsit, writing and formatting the video display, to screen editing, saving the file, and formatting and printing the document. Trivial, but it would make for a more useful chronological reference.

The only legitimate complaint I have deals with the omission of an easy reference sheet that lists in abbreviated form all the Scripsit commands and functions so you don't have to browse through the entire book for an obscure command.

Such a sheet would save a lot of anguish and time. Haga does, however, include an incomplete facsimile that lists only the screen format defaults. Useful, but not what I was looking for.

Using Scripsit is comprehensive, well-written, and useful. Every Model I/III owner with Scripsit, whether a first-time user or an old dog, can benefit and learn from this book.

It belongs next to your computer and its operating manual. *Using Scripsit* will soon become your most valuable guide. ■

★★★★★

CP/M 2.2, MBasic 80
on the Mapper I (48K)/III (64K)
Omikron Systems
1127 Hearst St.
Berkeley, CA 94702
Models I and III, 48K
One disk drive
Model I, $139
Model III, $199

by John B. Harrell III

Omikron Systems distributes hardware and software for TRS-80 computers that allow full implementation of the CP/M operating system—not another attempt to work around the TRS-80 internal memory structure. I used the Mapper and its software almost daily for three months. The system performed flawlessly—no glitches, surprises, or snags.

Because the hardware used in this modification is part of another review (see "CP/M III Ways," p. 122), I'll consider only the CP/M 2.2 operating system and the MBasic 80 interpreter distributed with the hardware modification as the starter system.

CP/M Operating System

CP/M isn't just another disk operating system; it's the most widely distributed operating system for the 8080 microprocessor. CP/M derivatives now exist for the 8086 family and the 68000 family of 16-bit computers. Radio Shack's introduction of the Model 4 with CP/M reinforces the fact that CP/M has become the de facto industry standard in operating systems.

The CP/M operating system is a monitor and control program for microcomputer systems that use floppy or hard disks for back-up storage. It is estimated to be in use in over half a million microcomputer systems throughout the world.

The system distributed with the Omikron Mapper I/III modification is CP/M 2.2. CP/M is logically divided into four parts: the Basic input/output system (BIOS), the Basic disk operating system (BDOS), the console command processor (CCP), and the transient program area (TPA).

These component parts provide features found in many other systems available for TRS-80s. For example, the BIOS provides system-dependent primitive operations necessary to access the disk drives and to interface with the standard system peripherals, BDOS is the disk management part of CP/M, and the CCP provides the interface between the operator and the system.

CP/M commands break down into three categories: built-in commands, transient commands, and user-implemented commands. See Table 1 for the commands in the first two groups (standard on CP/M system disks. The latter group are those commands programmed by the user in Assembly language and executed from the disk (similar to CMD files).

In addition to the standard Digital Research software and documentation, Omikron provides many specialized features particular to the TRS-80 implementation. This documentation is provided in *The Omikron Mapper Owner's Manual.* This contains all the instructions on the Mapper's installation and the documentation for the many special features required for TRS-80 systems.

One example of the customization Omikron provides is a sophisticated keyboard driver that maximizes the capabilities of the CP/M software interfaced with the TRS-80. Features include full keyboard debounce, a single-key control key, upper-/lowercase support with caps lock, and production of the full 128 characters in the ASCII set. Several of the keys have control functions for single-key ease.

Omikron provides serial and parallel printer drivers to support all printer installations. The default printer driver is parallel, supporting the majority of Radio Shack hardware configurations. The routine Serial.COM (provided by Omikron) establishes the serial driver and initializes the RS-232 interface to the default conditions established with the sense switches.

The video driver emulates a Soroc IQ120 video terminal on the TRS-80. It provides full upper-/lowercase support and maintains full graphics capabilities. Also, the video driver provides a routine that allows cursor addressing (positioning the cursor to a specific location on the screen).

Omikron provides two versions of SYSGEN and Format on the distribution disk. The manufacturer specially tailors a version of each utility to the 5¼- and 8-inch disks normally used with CP/M on the TRS-80s.

OMCOPY.COM is a disk copy routine provided by Omikron to allow more rapid back-ups. The standard routine, Peripheral Interchange Program (PIP), provides a method to copy all files from one disk to another. The operation is similar to the "back-up by class" LDOS uses or the "copy by file" NEWDOS80 uses.

To circumvent the delay inherent in this method, OMCOPY provides a track-by-track back-up of one disk to another. Options allow copying only user files, only system information, or the entire disk.

MEMTEST.COM and DSKTEST .COM evaluate your system's performance. The functions of each are obvious.

Setup.COM lets you customize the Omikron software according to your personal preferences and as certain software applications require. You can store the options changed by Setup permanently on the system disk.

TRSCPM.COM provides a mechanism to transfer files from TRSDOS formatted single-density disks to CP/M formatted disks. Both systems use a soft-sector disk; however, both sector lengths and directory structure differ. This program moves data from the TRSDOS-format system to the CP/M system.

This feature also allows moving Basic programs from one system to the other as long as you save the TRSDOS program to the disk in ASCII format. The program transfers the files sector by sector, copying extraneous information in some cases (the bytes past the TRSDOS end-of-file location, for instance).

In addition to the many utilities included on the system disk, you can see from Table 1 that the standard CP/M system disk includes a powerful context

Built-In Commands

Command	Description
ERA	Remove the specified file(s) from currently logged-in or specified disk.
DIR	List the names of all files in the specified directory on the console device.
REN	Rename the specified file to the designated name.
Save	Save the specified number of pages (256-byte blocks) to the file designated. Pages are taken from the Transient Program Area beginning at 100 hexadecimal (hex).
Type	Type (display) the contents of the specified ASCII file to the console device.
User	Allow specification of different user numbers for maintenance of separate files in the same directory.

Transient Commands

Command	Description
ASM	Load the CP/M assembler and assemble the specified program. The Assembly source code is assumed to be in Intel source mnemonics and the assembler produces 8080 machine language in the Intel hex format.
DDT	Allow dynamic interactive testing of programs while operating within the CP/M environment (DDT includes a limited disassembler).
Dump	Dump the contents of the specified file to the system console in hex form (displays the file's contents, representing each byte with ASCII letters for the hex codes).
ED	Allow the creation and editing/correction of ASCII character files in the CP/M environment.
Load	Load the assembled program in the Intel hex format and convert it to an executable machine code file.
MOVCPM	Allow reconfiguration of the CP/M system for any memory size.
PIP	Allow movement of files from one form of storage to another. You can specify many powerful parameters to control file movement.
STAT	Provide status and general purpose access to pertinent system and disk parameters from the CP/M command level.
Submit	Allow the submission of CP/M commands in a batch manner. This is similar to the TRSDOS Do command. Parameter substitution is allowed within Submit files. The compiled Submit file is executed after Submit completes. You can chain Submit commands.
SYSGEN	Allow the generation of a system disk by properly initializing the CP/M operating system on the disk.
XSUB	Extend the power of Submit to allow input to user programs as well as to the Console Command Processor.

Table 1. CP/M Version 2.2 Commands.

editor (ED), an Intel-compatible 8080 assembler (including the ability to perform conditional assembly), and a dynamic system debugging monitor (DDT).

Microsoft Basic 80

CP/M does not come with a Basic interpreter. On many systems, you have to add this feature by purchasing an interpreter compatible with the operating system. Owners of TRS-80s have been spoiled by having a high-quality Basic interpreter available at the flick of a switch.

The Omikron Mapper package comes with the industry-standard Microsoft interpreter in two versions for 8080/8085/Z80 microprocessors and CP/M, MBasic.COM 5.2, and OBasic.COM 4.51. OBasic is for those users who have software compatible with this interpreter and who don't wish to change to MBasic.

MBasic 5.2 is the most extensive and powerful Basic interpreter available for microprocessors. It meets the requirements for the Basic ANSI standard and supports many features not normally found in other Basic interpreters, such as complex string manipulation routines.

MBasic is similar in almost all respects to the Basic interpreter implemented on the Model 4. In many respects, this is the same Basic currently on the Models I and III.

The most significant difference from Model I/III Basic is in variable naming conventions. MBasic allows variables names to be of any character string (letters, numbers, and decimal point) up to 40 characters, whereas Model I/III Basic recognizes variables names with a maximum of two characters; the remainder of the variables name is insignificant.

In MBasic, you can't use reserved words as variables names but a variables name can contain a reserved word within it. This generally leads to a program error of some type in Model I/III Basic.

This feature causes most of the incompatibility between MBasic and Model I/III Basic. In MBasic, you must surround each reserved word with spaces or delimit it in some other manner. The internal structure of the tokenized programs is different.

In order to transfer Basic code from the Model I/III to the CP/M system, you must first write or edit the program so the interpreter can clearly identify the reserved words (contrary to the programming hints that obtain maximal speed from these Basic programs) and you must store the programs on the TRSDOS disk in ASCII format. This requires a significant programming conversion for those programs written without spaces in the lines.

"New" features of MBasic include the while/wend structure for program control, the ability to call a subroutine written in another language and pass arguments to it, chaining programs while preserving variables, erasing array variables under program control, renumbering program lines, and swapping variables in a single statement. These features are new to those who have used Model I/III Basic exclusively.

One significant feature implemented in MBasic is the Call subroutine. MBasic provides a linkage to external subroutines via a Call statement.

This subroutine linkage allows preparation of segments of code using an assembler or other language translator and calls it from MBasic with a sophisticated argument list. The calling linkage is compatible with the Microsoft compilers (Fortran, Cobol, and Basic), letting you compile complex routines for speed and use them directly from MBasic.

Conclusion

As anyone experienced with the CP/M system knows, its heart is the BIOS implementation of system-specific functions. Omikron has outdone itself with a superb BIOS module.

When I ran some benchmark tests on the MBasic interpreter, I was astounded to find that this interpreter outperforms its relative in the Model I TRS-80 mode. Operating the interpreter in a Model I with no speed-up, I obtained execution times indicating a significant increase in speed for equivalent routines (MBasic compared to NEWDOS80 Disk Basic).

When you consider that MBasic allows 40-character names and requires that reserved words have some type of delimiter around them, this is even more surprising.

As with any product, the Mapper software has some negative aspects. The distribution medium is the standard 5¼-inch disk. The CP/M software is written on this disk in 35-track format with 18 sectors (128 bytes) per track.

Due to the CP/M format, these disks

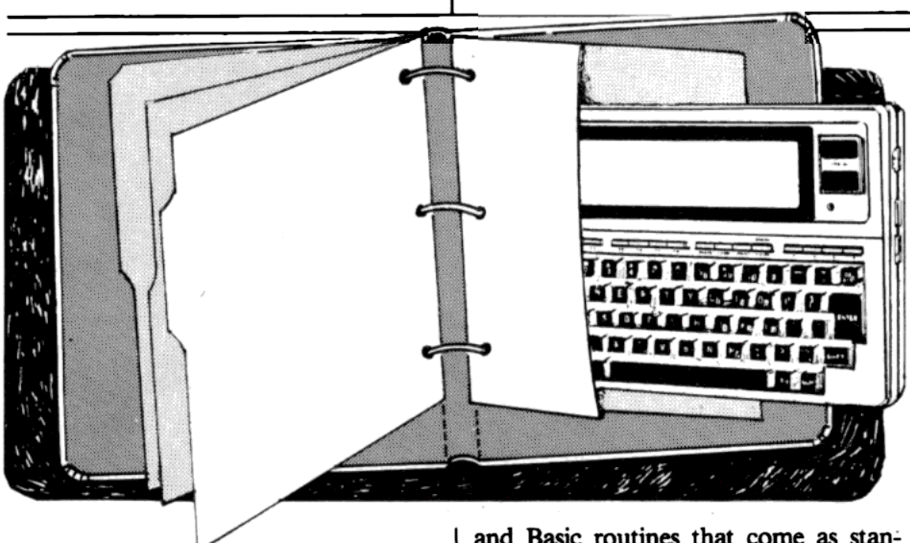
suffer a reduction of 10 percent capacity over the TRSDOS single-density format. After creating a 40-track system disk, working user storage is 81K (82,944 bytes out of 92,160 bytes due to system overhead). Compare this to the 102,400 bytes available on a 40-track, single-density TRSDOS disk.

Because of the limited storage available with the 5¼-inch disk, Omikron has to delete some of the standard files resident on CP/M system disks in order to include their specialized programs. These deleted routines are available to you at a nominal fee of $25 from Omikron and include items such as the Assembly source code to the BIOS and the Boot loader, MOVCPM.COM (relocates CP/M system BDOS and CCP for different memory configurations), and DEBLOCK.ASM (the source code for a CP/M disk deblocking algorithm).

Unlike Level II Basic, the manual supplied with MBasic doesn't offer the beginner an easy understanding of the language. Since this is not the sole source of Basic for your computer, this should be no problem.

Omikron now has a new software/hardware package available that further increases the TRS-80's power. At additional cost, they will provide TURBODOS and a replacement PROM chip for their Mapper installations.

Consider the wealth of software you obtain in addition to the hardware modification. CP/M 2.2 generally retails for approximately $150 and you must install the system (this usually means rewriting parts of the BIOS to add functions or change existing ones—no easy task for a beginner). MBasic usually retails for approximately $249. I cannot imagine where else you can get a bargain like this. ■



★★★★★

Businesspak +
Portable Computer Support Group
11035 Harry Hines Blvd. #207
Dallas, TX 75229
$89.95
SORT2 +
$29.95
Model 100
24K required
32K recommended

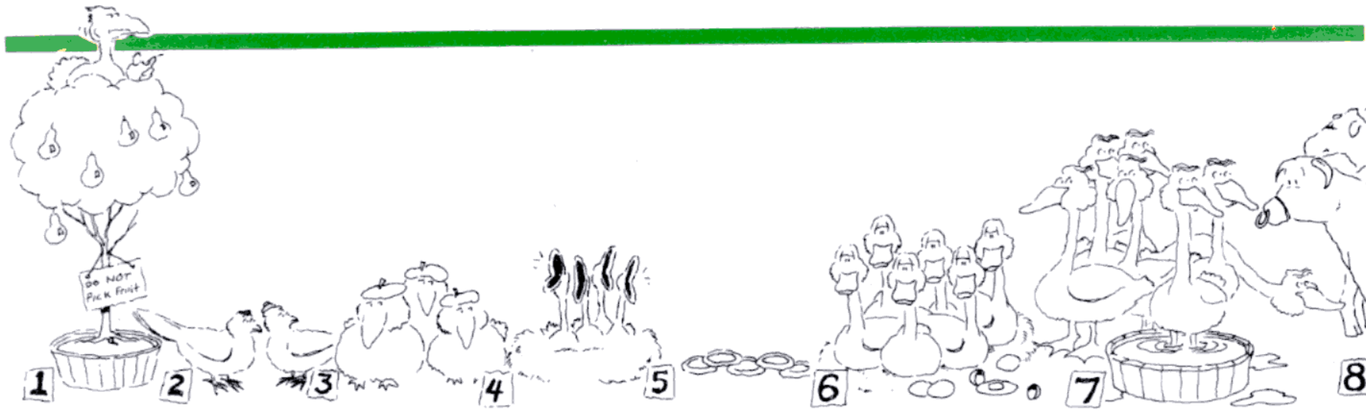
by Carl Oppedahl

Businesspak + is a nice collection of six programs for the Model 100. It augments the existing ROM-based word processing, telecommunications, and Basic routines that come as standard equipment with the Model 100.

Businesspak + contains Write +, a word processing program; EXPNS +, a simple spreadsheet program; Put +, a text entry program; Sort +, a routine that sorts records entered through Put +; Telex +, a package that lets you use Action Telex, a commercial telex service; and Graph +, a program that graphs data entered to EXPNS + on the Radio Shack DMP-100 printer.

Each program runs in Basic and relies occasionally on a machine-language routine hidden in memory. In addition, each relies on a corresponding do-file containing various operating parameters. You can change the do-file, often called a SPEC file in the documenta-

Continued on p. 69



Stretch the Twelve Days of

Give a gift subscription to **80 Micro** today and relax . . . you can't find a better gift for active computerists. Every issue gathers together the latest programs and projects for Radio Shack's personal computers, business computers, and portables. Your favorite computerist will be busy year round with:

Powerful Utilities—save time and effort and do more computing than ever before. Recent issues have included ways to improve Electric Pencil*, a program-to-program data transfer, and a cassette operating system.

Programming Techniques—learn how to program and spend less on software. Pick up a second language for special applications. Get faster results with machine-language subroutines . . . the monthly column "The Next Step" shows how.

Hi-res Graphics—make better business presentations with fancy printouts. Illustrate games with eye-catching details. Create unique 3-D art. It's all possible with articles on advanced graphics techniques.

Business Applications—increase office efficiency. Learn the business models' different capabilities. And keep abreast of developments for the Model 100* with reviews, applications, and utilities in "C•Notes."

Exciting Games—enjoy arcade games, adventure games, and simulations. Use the program listings to add features to other games. Or record high scores in "The Gamer's Cafe."

Two-way Communication—get definitive answers to technical questions in "Feedback Loop." Respond to articles, editorials, and news with "Input." Or submit an article for publication. **80 Micro** is a forum for ideas, not a one-way street.

"News" & "Remarks"—discover the latest industry trends and how they'll affect consumers. Learn about new services or novel applications. And get the inside story with publisher Wayne Green's outspoken editorials.

Home and Hobby—break the ice at a party with a singing computer. Or make electronics projects easier with a circuit-board scanner. Whatever the subject, there's always something of use in **80 Micro.**

Buyer's Guides—comparison-shop from home and save hundreds of dollars. Get all the facts on graphics software, printers, utilities, disk drives & disks, and more.

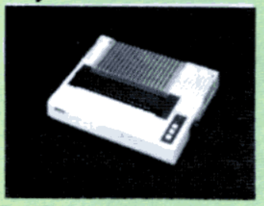
Candid Reviews—use "Reviews" and "Review Digest" to find out a product's strengths and weaknesses *before* it leaves the store.

80micro®

the magazine for TRS-80* users

EPSON

FX, RX & MX



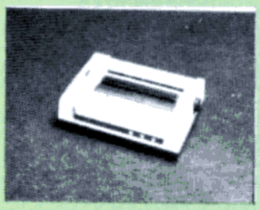
The **FX-80** features 160 cps, a correspondence font, 10, 12 & 17 cpi, italics, double-strike/width/emphasis, etc., dot graphics, friction/pin feed (the adjustable tractor is optional) & a 2K buffer. The 256 programmable characters use the 2K buffer space. The **FX-100** is the 136 column version & includes an adjustable tractor.

The **RX Series** replaces the MX, & offers 100 cps print speeds, but nothing more remarkable.

RX-80 .... **$399.88**
MX-80 F/T **$469.88**
MX-100 .... **$664.88**
FX-80 .... **$564.88**
FX-80 Tractor **$39.88**
FX-100 .... **$769.88**

C. ITOH

Prowriter

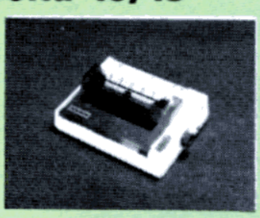


C. Itoh's venerable **Prowriter** has speed (120 cps), a buffer (1.5K), 10, 12, & 16 cpi (plus a proportional font with correspondance quality) and dot graphics (160x144 dpi). The **Prowriter 2** is the 136 column version.

Prowriter **$399.88**
Prowriter 2 .... **$719.88**

STAR MICRONICS

Gemini 10X/15 Delta 10/15



The **Gemini 10X** features 120 cps, 10, 12, 17 cpi, italics, a correspondance font, dot graphics & a 1K buffer. Friction/tractor feed. Use plain spool ribbons. The **Gemini 15** is the 132 column version. The **Delta 10** has all the features above plus parallel & serial interfaces, 160 cps print speed, an 8K buffer. The **Delta 15** is the 136 column version.

Gemini 10X **$309.88**
Gemini 15 .... **$459.88**
Delta 10 .... **$529.88**
Delta 15 .... **$CALL**

Smith-Corona

Memory Correct III Messenger

Here's the printer you've been waiting for. The Smith-Corona **Memory Correct III Messenger** is ideal for the home or small office. It combines the features of an electric typewriter ***and*** a letter-quality printer. And it's designed to handle both jobs with ease.

Features 12 cps, 3 pitches (10, 12 & 15), variable line spacing, 10.5" writing line, backspacing & auto-correction. Comes complete with parallel/serial interface.

Memory Correct III Messenger **$629.88**
TP-1/TP-2 .... **$CALL**

OKIDATA

Microline Series

The Microline 92 (80 col) & **93** (132 col) are ideal for word processing. They offer a 160 cps draft mode, a 40 cps correspondance mode, 10, 12 & 17 cpi (w/double-width), pin/friction feed (tractor is optional on the **92**) & dot-addressable graphics (120 x 1444). Centronics parallel interface is standard.

The **Microline 84** (132 col) is the Step 2 version, featuring 200 cps at 10, 12, & 17 cpi (w/double-width), all with a correspondance mode & dot addressable graphics. Parallel interface are standard issue.

The **Microline 82A** is a data cruncher, with 120 cps, 10 & 17 cpi, double-width, friction/pin feed on 80 columns. The **Microline 83A** is the 136 column version. Dot-addressable graphics are optional.

Microline 82A **$389.88**
82A/92 Tractor **$59.88**
Microline 83A **$599.88**
Okigraph I Dot Graphics
ROM (82A/83A) **$49.88**
Microline 92 .... **$459.88**
Microline 93 .... **$759.88**
Microline 84 .... **$1024.88**

MEMBER
DIRECT MAIL/MARKETING ASSOCIATION

MANNESMANN TALLY

MT-160 L

The **MT-160 L** (160 cps) has 8 fonts, parallel & serial interfaces, friction/tractor feed, & menu-driven installation from the control panel. The print quality is superior. The **MT-180 L** is the 136 column version.

A new, low cost draft printer, the **Sprite** (80 cps), is also available.

MT-160 L **$679.88**
MT-180 L **$849.88**
MT-Sprite **$329.88**

We sell other dot matrix printers, including the **Anadex WP-6000, IDS's Prism 80, Prism 132 & MicroPrism**, & the **Inforunner Riteman**. Call **(603) 881-9855** for technical details. For prices, or to order, call **(800) 343-0726**.

Letter-Quality Printers

C. ITOH

Starwriter

The C. Itoh **Starwriter** offers top speed (40 cps) at a good price. **Starwriters** use Diablo code, wheels & ribbons, 1/48" line space, 1/120" horizontal spacing—ideal for proportional modes. (For high speed output, there's the **Printmaster**, at 55 cps. Same specs as above.)

Starwriter Parallel **$1219.88**
Printmaster Parallel **$1679.88**

SILVER REED

EXP-550

The Silver Reed **EXP-550** is a 16 cps, 132 column letter-quality printer with *true* Diablo 1610/1620 emulation (sub/super scripts & underlining), making it compatible with most word processing software. It's ideal for medium duty office work. For light duty, there's the **EXP-500** at 12 cps (100 col).

EXP-550 (Parallel) **$719.88**
EXP-500 (Parallel) **$459.88**

We sell a variety of letter-quality printers, including the **Comrex ComRiter, Diablo 620 & 630**, the **NEC 2030, 3530 & 7730**, the **Qume Sprint 11+**.

STANDARD MICROSYSTEMS

Microterm

Microterm supports auto-dial/answer modems like the Hayes Smartmodem. Features pre-programmed dial & transmit, direct file transfer, 34K capture buffer and a certified 2400 baud operation rate. Specify Model I or Model III when ordering.

Microterm **$79.88**

DC Hayes Smartmodems

(300 baud) **$219.88**
(1200 baud) **$539.88**

Novation Modems

J-Cat (300 baud) **$139.88**
SmartCat (300 baud) **$199.88**
SmartCat (1200 baud) **$499.88**

Information/Orders:
(603) 881-9855
Orders Only: (800) 343-0726

No Hidden Charges:

We pay UPS ground shipping on all our orders, and we *never* charge extra for credit cards. We accept CODs & add a $10 fee per order. We have a $50 minimum order. Personal checks are cleared in 3 weeks.

All our equipment is shipped factoryfresh, with full warranties. We're authorized for warranty work on a number of printers, & we also offer extended warranty plans for those printers.

Sorry, we cannot accept open POs or extend credit/terms at these prices. APO and foreign orders are not accepted.

We prepared this ad in September & prices do change, so call to verify them.

Our **Computer Showroom** is now open in Amherst, NH.

MasterCard VISA

HIGH TECHNOLOGY AT AFFORDABLE PRICES

THE BOTTOM LINE

AMERICAN EXPRESS

MILFORD, NH 03055-0423 □ TELEPHONE (603) 881-9855 tion, to modify the behavior of the Basic program.

Write+ is a delightful program that lets you print out files created through Text. You can arrange to skip perforations, pause between pages for printing on single sheets, and print multiple copies, double- or triple-spaced output, and page headers and footers with the time and date.

The two types of word processors in the world are those that display on the screen exactly what will appear on the printed page, and those that display raw text on the screen and format the text for printing after you complete all your editing. Write+ is the latter type.

For instance, text that is double-spaced on paper appears single-spaced on the screen. Given that the Model 100 screen is much smaller than most CRTs, that is the best choice.

The Model 100 input and output routines are largely device-independent. Any of several devices with appropriate file names can be the object of Open and Close statements. These include RAM, LCD, CAS, MDM, LPT, and COM.

Write+ takes advantage of this. You can take the output that would usually go to the printer and route it to RAM for later editing or to COM for printing on a serial printer. Similarly, though you usually use Write+ to print files put in RAM by means of Text, it can take its input from any other device, such as a file on cassette.

Write+ is easy to use. You can easily print out text files created long ago with Text, skipping perforations, within 20 minutes of opening the Businesspak+ package.

Write+ uses simulated form feeds. When the time comes to skip forward to the top of the next page, the program uses individual line feed commands with an ASCII value of 10. Many printers, however, are capable of responding to a form feed character with an ASCII value of 12.

This accomplishes motion to the top of the next page much more quickly and quietly than a series of line feeds. It would have been nice if W+SPEC let the user select simulated or actual form feeds.

I have, however, determined a way to modify Write+ so that form feeds accomplish the perforation skip and blank page skips. The new lines appear in Program Listing 1.

EXPNS+ is an aid to travelers who must keep an expense record. Though it's described as a spreadsheet program, it is quite limited in capability. The program sets up an array of numbers, drawing up totals for up to 18 rows and 12 columns.

It does not let the user define relationships between and among different locations in the array, but merely adds up each row and column. Thus EXPNS+ is far less versatile than programs like VisiCalc.

"EXPNS+ is an aid to travelers who must keep an expense record."

You can store the numerical values in the array in RAM, then transfer them to and from other devices, such as a cassette. You can use the RAM file as input for the Graph+ program or merge it into a word processing file using Text.

Put+ is a simple data entry program. Given a user-defined P+SPEC file, the program accepts user keyboard entries and assembles uniform-length ASCII records composed of fields of user-determined fixed lengths. The program defines the fields within records in a SPEC file that the user creates. When you run the Put+ program again later, the program adds new records to the end of the existing file.

During keyboard entry of data, you go from one field to the next by means of the down-arrow key, and store the record by pushing the enter key. I found that to be awkward, and often pressed the enter key at the end of a line when the software expects the down-arrow key.

Also, when filling in, for example, a three-character field with a three-digit entry, I often found that Put+ jumped ahead to the next record when it should have simply accepted text for the next field in the same record. PCSG has since corrected both of these awkward situations in Put+.

Given the SPEC file and assuming you have fewer than 256 records, the Sort+ program then sorts the records as discussed below. You can also easily search the records using SCHEDL and ADDRSS (or even Text).

The Phone Line Hwy. 11 South Trenton, GA 1-404-657-6948

However, when you display the file on the screen or print it on the printer, it tends to look irregular. This occurs because the program runs adjacent fields together without spaces to separate them, and because both the screen display and print routines wraparound entire words at the end of the line rather than cut them off. According to PCSG, a new program called Data + ($59.95) is available that allows variable-format displays and listings.

The Sort + program allows sorting of up to 255 uniform-length ASCII records according to any selected constant-length field in the records. It's easiest to use Sort + with records created through Put +, but you can use Sort + on other files too, as long as the records and fields within records are of uniform length. Simply create a Put + specification file to match the fields in the file you want sorted.

Also, when Sort + reads in the file it will sort, it puts any lines of nonstandard length at the beginning of the output file. This is handy for two reasons: You can leave explanatory material, such as a heading or verbal summary, at the beginning of the file and it isn't disrupted by the sorting process.

Program	K to load	Free K to run
Write +	4.4	2
EXPNS +	7.0	5
Put +	2.8	1
Sort +	2.6	4 +
Telex +	3.9	1
Graph +	6.0	7

Table 1. Program sizes.

If you mistakenly include a line of inconsistent length in the file, it appears at the beginning as well. This brings the line to your attention so you can correct it.

The sort is a simple sort in ASCII order. The documentation claims that you can sort 255 records in 15 to 20 seconds. One file I sorted actually required just over two minutes. I attribute this to the fact that Sort + swaps entire records whenever a comparison so indicates.

This isn't really a drawback, but a consequence of the Model 100's limited memory. On a larger machine, it would be possible to sort more efficiently by extracting and sorting key fields, and by swapping the entire records only when the key fields are in order.

PCSG has also released another program, Sort2 +, that does a few things Sort + cannot. For example, it sorts true numeric data so that 6 comes out above 10 in order.

Or you can set Sort2 + to sort with the uppercase flag bit suppressed: The result is that the program treats upper- and lowercase letters equally in the sorting process. Finally, it performs the sort directly on the RAM file, rather than by reading the file into the Basic program area. Sort2 + needs less free RAM than Sort +.

Telex + helps you send telexes, mailgrams, and the like from text files you've previously typed in and saved. To use Telex +, you must open an account with Action Telex, a company based in Dallas that accepts messages through a modem over phone lines and sends them out via the Western Union telex network.

This requires payment of a $150 annual registration fee and a moderate usage charge. The advertisements I've seen for Businesspak + don't make this point very clear. The Businesspak + documentation describes a free trial arrangement whereby you can send three free telexes or mailgrams.

I tried this by calling the phone number in the manual and supplying the requested information. I was told that Action Telex would contact me with an identification code for the free trial. Three weeks and three more phone calls passed before I finally got a code.

When I tried it out, the program indicated that Action Telex had accepted my mailgram and two telexes for delivery, but none of them reached their destination. When the local office finally called with another code, I had better luck. The mailgram got through all right, but the telex was delayed eight hours.

Though the documentation said nothing on the matter, I found I had to precede the destination telex number with a zero. After that I had no trouble with Action Telex.

I also had some problems with the Telex + program itself. The Model 100 has a built-in modem with provisions for acoustic coupling and direct-connect autodial operation. However, one constant feature of the program is that it tries to dial the phone number for access to the Action Telex computer whenever you run it.

If you are using the acoustic coupler, this doesn't hurt anything, just wastes time. But if you are trying to dial the access number manually for MCI access or to get through a Touch-Tone-only switchboard, the dialing routine frustrates your efforts by hanging up the phone for a second before listening for the carrier tone.

It would be helpful if PCSG modified Telex + to allow the equivalent of the Term key in TELCOM, and thus bypassed the autodialing routine. One remedy is to use a duplex jack instead of the silver phone cord. Plug in both the phone and beige cable. Do not hang up the phone until the Model 100 detects the carrier.

Graph + lets you prepare graphs based on numerical and text information typed in using the EXPNS + program. The graphs can be pie, bar, and line charts.

Since Graph + takes its input only from EXPNS +, and since EXPNS +

For Version 1.3:

```
1400 IF FF%<>0 THEN PRINT#2,MID$(LF$,1,2*(BM% + LC%));: HF% = 2:GOSUB 1550
1405 PRINT#2,CHR$(12);:IF BL% – PP%>0 AND PG%<PE% THEN 1490
1410 IF CN%<NC% THEN CLOSE:BL% = 160:PP% = 0:GOSUB 1450:GOTO 130 ELSE
PRINT#2,CHR$(12);
```

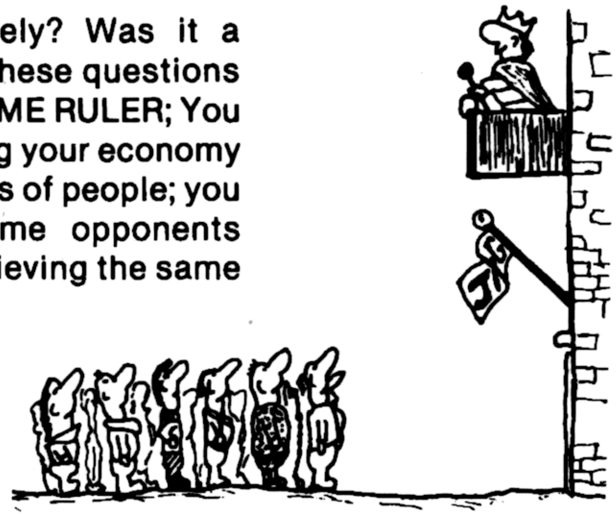
For Version 2.1:

```
32 IFFF<>0THENPRINT#2,LEFT$ (LF$,2*(BM + LC));:HF = 2:GOSUB43
33 PRINT#2,CHR$(12);:IFBL – PP>0ANDPG<PETHEN39
34 IFCN<NCTHENCLOSE:BL = 160 :PP = 0:GOSUB37:GOTO6
ELSEIFFDTHENPRINT#2,CHR$(12);
```

Program Listing 1. Form feed modifications for Write +.

SUPREME RULER

Have you conquered the World lately? Was it a Challenge? If the answer to either of these questions is NO, you obviously don't have SUPREME RULER; You haven't led your own country, managing your economy and controlling the destinies of millions of people; you haven't had the chance to overcome opponents (Human and/or Computer) bent on achieving the same goals.



In SUPREME RULER you must protect your economy (or else risk a recession), support and feed your people, encourage new business, manage government revenues, and control your army. These and many other factors are interdependent, making this a very challenging game/simulation.
Supports up to 9 countries (4 with 16K), either human or computer controlled. The computer plays by all the rules, and makes a worthy opponent in single-player games.

16K and 32K versions together on Cassette, with 40 Page Manual.	PRICE:	$18.50
32K version (with SAVE GAME feature) on Disk with 40 Page Manual.	PRICE:	$20.50

Model I/III

For those of you with 48K:

SUPREME RULER PLUS!

The extra memory is filled with many new features, expanded reports, more intelligent computer opponents with selectable "skill levels", and MUCH MORE! Comes with a 50 page manual and a start-up booklet.

Provided on either Disk or Cassette. PRICE: $26.50

DEALER INQUIRIES INVITED

OUR "TRIPLE PROTECTION" GUARANTEE

In today's software world it is hard to tell a good program from a bad one by only a magazine ad; that's why JMG has a warranty plan that will guarantee your satisfaction. Here is our "Triple Protection" guarantee:

1) **QUALITY GUARANTEE:** If you are unsatisfied with the software product you have purchased, you may return it (in good condition) within 14 days for a refund, less a $2 handling charge. (We also ask that you send us a brief letter stating the reason for your return.)
2) **UPDATES GUARANTEE:** We always listen to our customers, and we often improve and expand our software products on the basis of their comments. If we make an improvement to a product and release a new version, all our previous purchasers will be notified and offered the update. To receive an update, you only have to return your original disk plus $1 for shipping costs.
3) **MEDIA GUARANTEE:** The diskette or cassette has a Lifetime Guarantee. If yours turns out to be defective, or it fails and you do not have any backups made, then just return the disk/cassette and we'll replace it.

TO ORDER: Mail or Phone your order,

OR

Go and see your nearest dealer. If he doesn't carry JMG programs, shout at him a bit.

JMG SOFTWARE INTERNATIONAL

P.O. BOX 598
FALLS STATION,
NIAGARA FALLS, N.Y.
U.S.A. 14303

OR

710 UPPER JAMES ST.
HAMILTON, ONTARIO
CANADA L9C 2Z8
(416) 389-6086

THE FINE PRINT:

We accept VISA and MasterCard.
All prices in U.S. Funds.
Please allow 1 or 2 weeks for PERSONAL CHECKS to clear.
SHIPPING CHARGES are $2.00 in North America, $5.00 overseas.

EXTERMINATE!

The World is counting on you to annihilate a horde of alien bugs that are trying to escape from an underground cavern. Your only weapons are your skill, strategy, and an armed "Rover". This 100% Machine Language program has quality graphics and sound, and surprises to keep you busy at every new skill level.

EXTERMINATE on Cassette,	PRICE:	$15.50
Model I/III Diskette,	PRICE:	$17.50

✱ Order now and you will receive a free "Bug Hopper" Jumping Disc! ✱

126

handles only up to 18 columns, that's the largest number of points you can graph. Thus any line graph is, of necessity, quite bumpy.

The Businesspak + software comes on cassette, all of which you can back up. I found the cassettes themselves to be of high quality and well recorded; I had no load errors after dozens of loads. You must load each program from cassette before use, which takes as long as four minutes.

Documentation and Support

The documentation, totaling some 108 pages, is written under the assumption that the user is already familiar with the connections to the Model 100, has mastered the built-in program Text, and has at least passing familiarity with the other built-in software.

The documentation has no general index, but you don't need one since index tabs for each program allow easy reference to explanatory material. In addition, Write +, perhaps the most complex program, has its own index in the most recently received documentation.

The PCSG telephone number appears prominently on the manuals and instructions. I called several times at various times of the day and night, and always reached cheerful people. Though I had difficulty getting signed up with Action Telex, the frustration was outweighed by the many other times that my questions were answered immediately and correctly.

Each time I called, PCSG asked for the serial number of my software package. I consider this a perfectly reasonable request as it helps protect against software pirates, but the drawback was that I was immediately identified as a reviewer. Nonetheless, I sense that all purchasers can expect good customer support.

User-Friendliness

The programs are well error-trapped, although occasionally I tripped them up with unusual inputs. For example, each program depends on one or more SPEC files containing file formats, field names, and so on. If the SPEC file is not set up properly, the program using it can go astray. I had particular problems coming up with a P + SPEC file suitable for use with Put +.

"Businesspak + provides a good package of routines for the Model 100 owner. They are easy to learn and use."

One problem anyone faces who writes text-handling programs for the Model 100 is that RAM files can have arbitrarily long records. A Line Input or Input that goes to a string variable can cause an OS (out of string space) error that's hard to protect against.

To alleviate this problem, Microsoft provided the INPUT$ function, which gets a specified number of characters from an input device regardless of line terminators. Even that function generates errors; for example, if three characters are requested, and if only two characters are left in the device you'll get an end-of-file error.

The writers of Businesspak + use a machine-code routine of about 166 bytes called from Basic. This routine goes to the input device and gets the next printable line of characters up to the next carriage return or space.

One drawback of the machine-code routine is that if no space occurs within the next 160 characters, the routine only returns a single character. I discovered this when I attempted to list a Basic program with all the spaces removed. Some of the long lines in the file printed out in a column, one character at a time. I had to go back to the Basic program and insert spaces to make listing through Write + possible.

The programs are written in Basic rather than machine code. The advantage of Basic is that the user can easily modify the code, perhaps to customize it for a different printer or add a feature. The disadvantage is that machine code can be faster and more compact. For most of the programs, such as Write + and Put +, the execution, printing, and keyboard delays are not even perceptible.

Even if you have the maximum RAM available to the Model 100, 32K, you don't have enough room to leave all the Businesspak + programs in RAM all the time. The instruction book suggests that you purge the programs when you are done and reload them later from tape. Personally, I find Write + so handy that I have left it in RAM continuously since first loading it.

The advertising and the manual I initially received are silent on the required RAM. Most of the routines fit in less than 4K, leaving a little work space for an 8K user (see Table 1). No serious user should consider less than 24K, and I recommend getting a full 32K.

Businesspak + provides a good package of routines for the Model 100 owner. They are easy to learn and use. The graphing program works only on the Radio Shack DMP-100, and the telex program involves a further expenditure of $150 per year. But the word processing, sorting, text entry, and simple spreadsheet programs are general in their scope and utility. I recommend them. ■

Assembly Language Made Simple—Part I

by Hardin Brothers

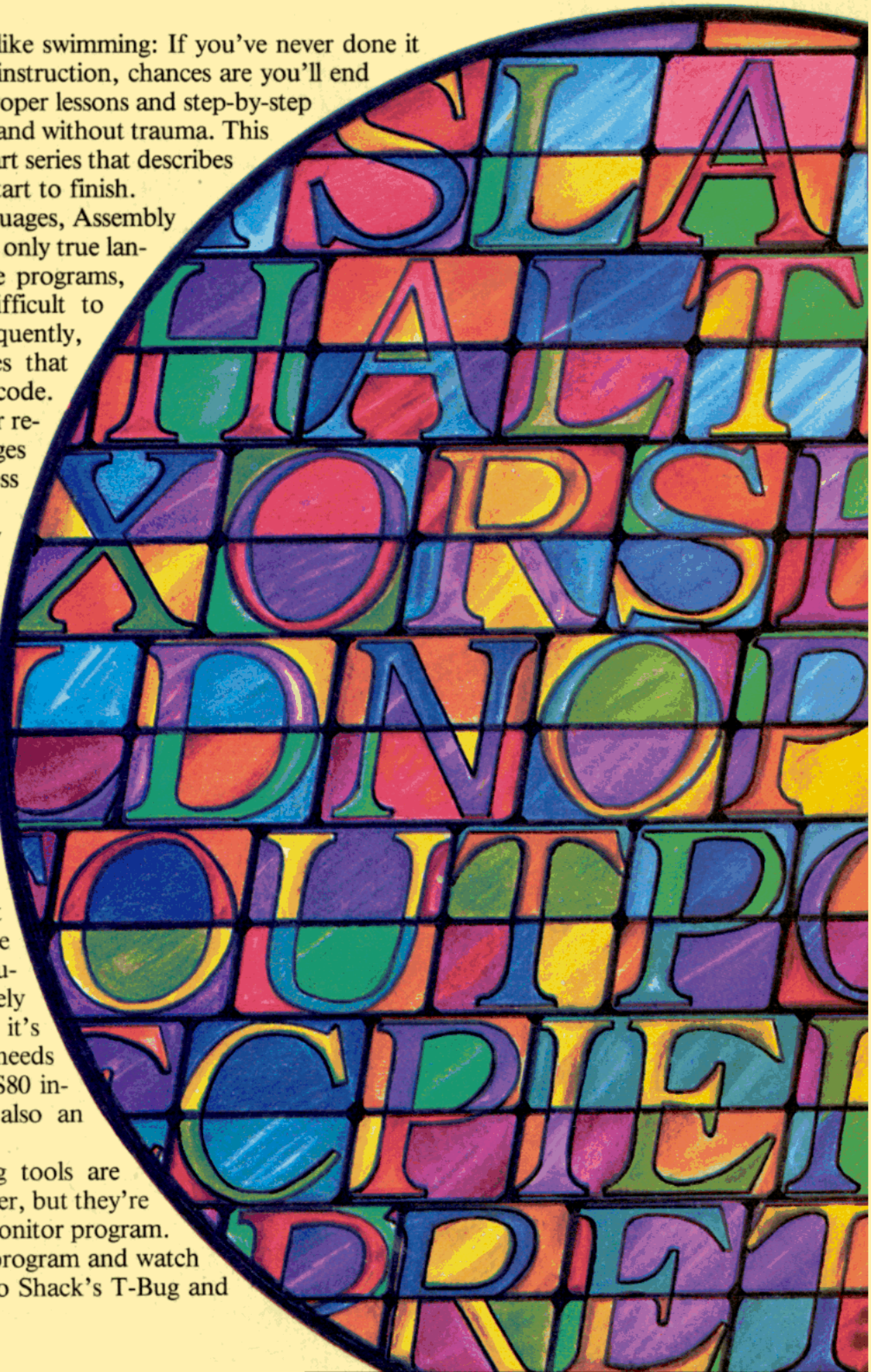
Assembly-language programming is like swimming: If you've never done it before and plunge in without any instruction, chances are you'll end up over your head. However, with the proper lessons and step-by-step guidelines, you'll learn the skill smoothly and without trauma. This article is the first in a consecutive three-part series that describes Assembly-language programming from start to finish.

Like Basic and all other computer languages, Assembly language is a variation on the computer's only true language, machine code. Machine-language programs, comprising only 1's and zeros, are difficult to write and prone to typing errors. Consequently, programmers developed other languages that use mnemonic code to replace machine code. Although easier to work with, the further removed from machine code these languages are, the slower they execute and the less powerful they are.

Assembly language provides a happy medium. It combines the advantages of simple code with the power of a language that closely resembles machine code. As a result, an Assembly-language program runs faster and provides more computational power than, say, a Basic program.

Before you begin to write in Assembly language, you need the proper tools. Most important is a program called an editor/assembler. If you don't own one, the Radio Shack Series 1 Editor/Assembler is a good choice to get you started. It's powerful for the price ($30 to $35), comes with excellent documentation, and while it doesn't completely satisfy an experienced programmer, it's powerful enough to satisfy all your needs until you become an expert. NEWDOS80 includes Apparat's EDTASM module, also an excellent first assembler.

Several other beginner programming tools are helpful in addition to the editor/assembler, but they're not mandatory. Most useful is a good monitor program. Monitors let you single-step through a program and watch the execution of each instruction. Radio Shack's T-Bug and



Here's your first lesson in Assembly-language programming. This article describes how the Z80 chip processes Assembly instructions.

Debug, as well as any DOS Debug, can start you off, but more powerful monitors are available.

Other useful tools are a disassembler program, Nanos Systems Corp.'s Z80 reference card, a printer for listings (invaluable for complex debugging), several reference books, and a disk system to simplify loading, saving, and assembling programs. These are all extras that make Assembly-language programming easier; at first, all you need is an editor/assembler.

The Z80

The heart of your Model I, II, III, or 4 is a Z80 microprocessor chip (or Z80A, the same thing but faster). Though this is one of the most advanced 8-bit microprocessors made, with 22 registers and a strong instruction set, it still has a very limited repertoire of capabilities.

The Z80 can get a value from memory, put a value into memory, move values around in between its internal registers, add and subtract two values, perform simple logical manipulations (AND, OR, and Exclusive-Or), report some information about the results of its operations, keep track of where to find its next instruction, and manipulate a simple data structure called a stack. It can't multiply or divide, directly print to the screen or printer, or do most of the things your computer does in Basic.

Your job as an Assembly-language programmer is to utilize the simple abilities of the Z80 chip to develop a complex program, usually complete with input from keyboard, tape, or disk and output to tape, disk, screen, or printer. To do so, learn to think in terms of the Z80's capabilities, which means you have to break each operation into specific, small steps. If you can train yourself to analyze and build programs in these small steps, you're well on your way to becoming an Assembly-language programmer.

Assembly and Machine Language

The only instructions the Z80 understands are sequences of 5-volt and zero-volt electrical impulses. Generally we refer to those impulses as 1's and zeros.

For example, the sequence of impulses represented as 11001011 00100111 makes the Z80 copy a specific single bit from one place inside itself to another. Because sequences of 1's and zeros look like a binary number, programmers often think of them as such, and translate them into hexadecimal or, rarely, decimal numbers. The above instruction would thus appear as CBH 47H in hexadecimal notation, 203 071 in decimal. When you write programs by loading such values directly into memory, either with POKEs or by using a monitor program, you program in the Z80 native tongue, machine language.

However, the Z80 recognizes almost 700 such instructions. Except in unusual circumstances, it's a waste of time for a programmer to look up each instruction in a table and enter its special sequence directly into memory. That's why the first computer language ever developed was "Assembler." Using an assembler, you can write instructions using mnemonic (memory aids), 2- to 4-letter abbreviations of program instructions. You can write the binary instruction above as BIT 0,A which, while still far removed from English, is much easier to remember and understand.

The assembler changes the mnemonic into the appropriate bit sequence by looking it up in a table. The list of mnemonic instructions, which you write, is called a source code; the list of machine instructions, which the assembler writes from the source code, is the object code. The Z80 chip cannot act directly on source code instructions, so the translation into bit sequences is necessary before you can run your program.

Unlike Basic, assemblers are not interactive. You must write the source code, assemble it into machine language (which you save on either tape or disk), then load the object code back into the computer to run it. If a bug occurs in the program (and one almost always does), you must then reload the assembler program, reload your source code, correct the source code, save your source code back to tape or disk, assemble the program again on tape or disk, load the new object code back into memory, and run it again.

Virtually all assemblers come with an editor, a separate but related program that helps you write and correct source code. The editor is a simple, line-oriented text processor that includes the ability to number lines of source code and to find lines by reference to their numbers.

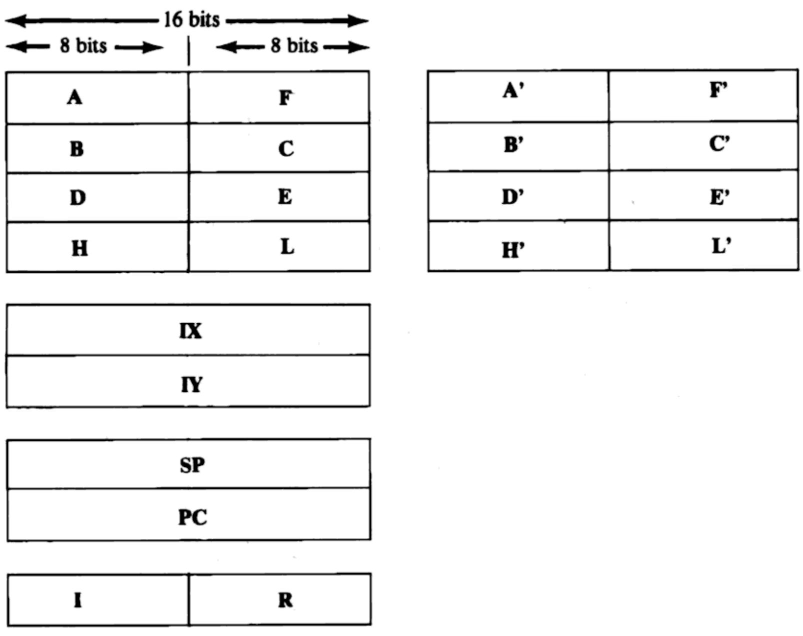


Figure 1. The Z80 registers.

Whenever you use an editor/assembler package to write a program, you must go through a four-step process:

- Use the editor to write the source code.
- Save the source code to tape or disk.
- Assemble the source code into object code and save the object code to tape or disk.
- Leave the editor/assembler program, return to either Level II Basic or DOS, load in the object code form of the program, and run it.

The finished program probably loads as a SYSTEM program, if it's on tape, or as a /CMD program if it's on disk.

Memory and Registers

Your computer provides several different types of memory. Basic resides in ROM, the keyboard and screen each have specially dedicated random-access memory (RAM), and programs and data use between 16K and 48K (on the Models I and III) of general-purpose RAM. Each byte of memory has a unique address which the Z80 uses to find stored values and to place values in the correct locations.

Like program instructions, memory addresses are nothing but unique sequences of electronic impulses, but it is easier to think of them in terms of hexadecimal or decimal numbers. You can remember memory addresses more easily in hexadecimal once you become familiar with that numbering system.

In Models I and III, ROM occupies addresses from 0000 hexadecimal (hex) to 37FF hex, the keyboard from 3800 hex to 3BFF hex, the video screen from 3C00 hex to 3FFF hex, and general memory from 4000 hex to 7FFF hex, BFFF hex, or FFFF hex depending on the amount of RAM in your system. All of the programs I include with this article run with any size RAM (except 4K) for a Model I, III, or 4 (in the Model III mode).

You may notice that memory holds both program instructions and data, and wonder how the Z80 distinguishes the two. It doesn't. As far as the Z80 is concerned, no difference exists between the two; you and your program decide which parts of memory hold instructions and which hold data. However, if you make a mistake and allow the Z80 to operate on data as if it were instructions, you face the infamous "silent death"—either a locked up computer or a spontaneous reboot.

The Z80 contains 208 bits of memory organized into registers (see Fig. 1). Each register is either 8 or 16 bits wide—that is, each holds either 8 or 16 bits of information. As a programmer, you manipulate the values held in these registers, and copy information from the registers to RAM and vice versa.

The A register, which holds 8 bits, is the accumulator. It's a general pipeline for moving data into and out of the Z80 chip, and also aids in almost all of the arithmetic and logical operations the Z80 performs.

The individual bits of a second register, called the F or flag register, holds information about the results of various internal operations. The Z80 uses these flags to perform conditional branches (similar to the Basic If...Then command).

B, C, D, E, H, and L are all general-purpose 8-bit registers that hold temporary data and operands for arithmetic and logical operations. You can also use these registers in pairs: BC, DE, and HL. Each pair holds 16 bits and can hold data addresses in memory, or large values used in some arithmetic functions.

The HL register pair also has a special function. Whenever the Z80 performs 16-bit arithmetic operations, the HL register pair becomes an accumulator and holds the results of the calculations. Using any register pair as two 8-bit or one 16-bit register is entirely up to you, and you can change their function at any time.

Two 16-bit index registers exist, IX and IY. They almost always hold addresses of data tables in memory, and make accessing that much easier.

SP, the stack pointer, is a special-purpose register that holds the address of (points to) a data table called the stack. Stack operations let the Z80 return to the appropriate location after

```
00100 ;*********************************
00110 ;*                               *
00120 ;*   The First Program  --       *
00130 ;*                               *
00140 ;*   Print "HI" in the upper-    *
00150 ;*   left corner of the screen   *
00160 ;*                               *
00170 ;*********************************
00180 ;
00190         ORG     7000H             ;Define beg. of prog.
00200 ;
00210 START   LD      A,48H             ;48H is ASCII value of H
00220         LD      (3C00H),A         ; and put it on screen
00230         LD      A,49H             ;49H is ASCII value of I
00240         LD      (3C01H),A         ; and put it on screen
00250 LOOP    JP      LOOP              ;Loop forever
00260 ;
00270         END     START
00280 ;
```

Program Listing 1. Assembly-Language Lesson 1.

The Key Box

Model I and III
16K RAM
Assembly Language
Editor/Assembler

Shape your TRS-80 to communicate with any computer you want.

Omniterm is the most flexible, powerful terminal program you can buy. Omniterm lets you adapt your TRS-80 to communicate with 99.9% of the world's computers. Your company's mainframe, for example. Or any other personal computer, time-sharing computer, or communications service.

Omniterm overcomes incompatibilities in screen formats, baud rates, character sets, control codes and file transfer protocols. Seven complete translation tables let you change any character, for complete compatibility of all input and output devices. Omniterm is so flexible, users have even set up their ASCII-coded systems to communicate with EBCDIC-coded systems.

You can send all ASCII characters, even those that aren't on your keyboard. Reformat your screen to neatly accommodate any line length. Run your printer while you're sending or receiving data. And even review data that's scrolled off the top of the screen.

Omniterm's well-thought-out design makes it easy to use. You can get a status display of all functions while on line to tell you what's going on, and make any changes at the same time. You can create a special file of your settings to make it easier next time. You also get X/Y cursor control, single keystroke sign-on and auto-dialing. Even a phone directory. And lots more.

You don't have to be a computer expert to use Omniterm. Just spend a day with what the reviewers call "the best manual in the business." Then if you need help, just call, write, or contact us via CompuServe, Delphi, or Source.

Omniterm is the proven terminal program. The program thousands of people have used successfully. And the one the editors call the "top program available" (Byte, 80-Micro, Infoworld, etc.)

Omniterm comes complete with sample setting files, conversion utilities, a practical text editor, seven translation tables, and a 76-page manual with index.

Available at leading dealers, or prompt shipment on direct phone and mail orders. Order Omniterm, for a super-smart TRS-80 that's putty in your hands.

Only $95 for TRS-80 Models I, III or 4 (32K memory, one disk minimum). $175 for Models II, 12 or 16 (64K memory minimum). (In Mass., add 5% sales tax.) MasterCard, VISA, and C.O.D.

IBM PC version coming soon.

DEALER INQUIRIES INVITED.

Telephone: (617) 852-0233
CompuServe: 70310,267
Source: TCA818
Delphi: Lindbergh

NOW Version 1.60 with auto-dial and log-on direct from DOS!
Plus other new features.

Omniterm

Lindbergh Systems

49 Beechmont St., Worcester, MA 01609

135

See List of Advertisers on Page 291

it completes a subroutine, simplify saving data from registers temporarily, and allow transfer of 16 bits of information from one register pair to another.

The program counter, register PC, always holds the address of the next program instruction. The Z80 uses the PC register to keep its place in the program while it carries out an instruction.

A', F', B', C', D', E', H', and L' (the prime set) are a second set of general-purpose registers. If you give the correct instructions to the Z80, it saves the regular general registers (A to L) and the prime set becomes active. A later exchange instruction switches the first set of registers back into active service. It's up to the program (and programmer) to keep track of which set is in use at any given time—the Z80 considers the inactive set the prime set.

Finally, there are two 8-bit registers, I and R, that programmers rarely use. The R register holds the address of the next bank of memory that needs refreshing (a kind of electronic tickling to stay awake). The Z80 handles the memory refresh chores automatically. The I register is useful for some interrupt processing, but the Models I and III don't include the hardware necessary to use it.

A First Program

The best way to understand Assembly-language programming is to start programming. Because every editor/assembler package includes its own unique editor instructions, I will not give explicit instructions on using the editor portion of your editor/assembler. Load the program and run it, then use the documentation to experiment with the editor until you are comfortable with its commands. You probably use a command like I to start entering text on numbered lines, a Break to return to the function prompt, a P to list what you have written, and so on.

Your first Basic program was probably something like

```
10 PRINT "HI"
```

so I'll start you with a similar Assembly-language program (see Program Listing 1). Even though this is a simple, short program, there is much to learn from it. The first nine lines, 100–180, are all remarks and the assembler ignores them. In most assembler formats, a semicolon precedes all remarks.

It is wise to liberally include remark statements in your source code programs—the remarks don't take up memory space in your completed machine-language program and they help you understand program logic several weeks later.

After the remark lines, everything in the program falls into four neat columns. The first column, or field, contains a five-digit line number as supplied by the editor. As I explained earlier, these numbers are used only by the editor and ignored by the assembler. They help you find the correct line to edit and the right place to insert new lines. Just as in Basic, it is normal in Assembly language to increment lines by 10 so you can insert lines later. Also, your editor probably includes a simple renumbering facility.

The second column contains labels, each of which can contain one to six alphanumeric characters. Labels can represent either values (as I show in the next program) or the addresses. The labels in this program, Start and Loop, represent the addresses of the instructions that follow them.

The third column contains either the mnemonics for the machine-code instructions or pseudo-ops—direct instructions to the assembler program that don't get translated into machine instructions. Simple assemblers provide only a handful of pseudo-ops that are easy to understand and use.

The fourth column contains the operands for each instruction. Each mne-

Making Sense of Those Crazy Numbers

by Amee Eisenberg
***80 Micro* Technical Editor**

The Decimal System

For those of us with 10 fingers, a decimal number system (based on units of 10) has always made intrinsic sense. You start at your thumb and keep counting until you run out of fingers. Then you make a mark to signify one group of 10 and start at your thumb again. Things are counted in groups of 10s plus any left-over 1's.

We write numbers using a place value system, placing units, or 1's, furthest to the right in a number. The next place holds the 10s, and the next the 10 times 10s, or 100s. Reading from right to left, the value of the place increases by a multiple of the base. Thus, 1, 10, and 100 signify one unit, 10 units, and 10 times 10, or 100, units respectively. You can represent this as follows:

```
Y Y Y    (* 1)
         (* 10)
         (* 100)
```

The Binary System

Your computer is based on a binary numbering system; that is, it uses a two-unit counting system. It counts with electrical impulses that are either off or on, which we represent numerically as 0 and 1. It also uses a place value system to keep track of larger numbers.

In a binary, or base 2, numbering system, the places hold multiples of 2. So the first place represents units, the next 2's, the next 4's. Thus, 1, 10, and 100 signify 1 unit, 2 units, and 2 times 2, or 4, units respectively. This is represented as:

```
Y Y Y    (* 1)
         (* 2)
         (* 4)
```

Binary becomes ungainly for humans as the numbers get larger. For instance, the number 136 decimal is written 10001000 in binary. That is, there are no 1's, no 2's, no 4's, one 8, no 16s, no 32s, no 64s, and one 128. Adding the one 8 and the one 128 results in the decimal equivalent, 136, as below:

```
1 0 0 0 1 0 0 0      0  (0 *  1)
                     0  (0 *  2)
                     0  (0 *  4)
                     8  (1 *  8)
                     0  (0 * 16)
                     0  (0 * 32)
                     0  (0 * 64)
                   128  (1 *128)
                   ---
                   136  decimal
```

Since humans count in decimal and computers in binary, a compromise counting system is necessary to make conversations between humans and computers a little simpler.

monic requires that zero, one, or two *operands* follow it. In most cases, the number of operands is obvious—you can't ask the Z80 to load (LD) a value unless it knows where the value is coming from and where it is going. So, two operands always follow the Load command.

Finally, Assembly code reserves the last column of each line for remarks. You can quickly learn to write cogent remarks that fit on the same screen line as the rest of the instructions and produce clean, easy-to-read source code.

Refer back to Listing 1. Lines 100–180 are remarks ignored by the assembler. Line 190 starts with the pseudo-op ORG, which defines the starting address of the program so the assembler can calculate addresses of each of the instructions.

Notice that the program's address, 7000H, is a hexadecimal number followed by the letter H. The same address, in decimal, is 28672. Unless a suffix of H or B follows a number, the assembler assumes that number is in decimal format. However, the assembler accepts a suffix of D to indicate a decimal number. Get into the habit of adding a base suffix to the end of every number you use in a source code, regardless of its base; it makes debugging much easier.

Enter the base 16, or hexadecimal, numbering system.

The Hexadecimal System

Hexadecimal (usually abbreviated hex), is just like decimal if you have 16 fingers. To count in hex, you say 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. A is equivalent to 10 decimal, B to 11 decimal, C to 12, D to 13, E to 14, and F to 15. Ten hex is equal to 16 decimal because, just like decimal and binary, hex uses a place value system to represent numbers. Ten hex means that there are no units and one 16:

```
Y Y   (*1)
      (*16)
```

When you write a hexadecimal number, we append the uppercase letter H to it to avoid confusing it with binary or decimal. Additionally, we write the numbers A through F as 0AH, 0BH, 0CH, etc., the zero eliminating confusing the numeral with a letter.

To understand why hex is convenient for the computer, you need to think about how the computer stores information. The Z80 microprocessor uses 8-bit logic. A bit is an electronic signal that's either on or off (binary, remember?).

A group of 8 bits of information clustered together is called a byte. A single hex numeral can represent half-bytes, or nibbles, consisting of 4 bits because the greatest value a four-digit binary number represents is 16 decimal. Remember, 1000 in binary is 10 in hex and 16 in decimal. So two hexadecimal digits can represent any 8-bit value (a byte).

It's easy to convert a binary byte to a hexadecimal byte. Take the number 10001000 binary again. The first step is to break the byte into its component nibbles, 1000 and 1000. Then convert each nibble to hex:

```
1 0 0 0   0  (0*1)    1 0 0 0   0  (0*1)
          0  (0*2)              0  (0*2)
          0  (0*4)              0  (0*4)
          8  (1*8)              8  (1*8)
          —                     —
          8H                    8H = 88H
```

Therefore, 10001000 binary is 88 hex. Check this by converting both binary and hex values back to decimal. Earlier I said that 10001000 binary is 136 decimal. 88H means 8 units plus 8 sixteens (or 128) which equals 136 decimal.

Try another binary-to-hex conversion, this time with 10011110. Break it into two nibbles: 1001 and 1110. Convert each nibble:

```
1 0 0 1   1  (1*1)    1 1 1 0   0  (0*1)
          0  (0*2)              2  (1*2)
          0  (0*4)              4  (1*4)
          8  (1*8)              8  (1*8)
          —                     —
          9                     E  (14 decimal)
                                =  9EH
```

So, 10011110 is 9E hex. If you check this by converting to decimal you find 10011110 equals $(128+16+8+4+2)$ or 158, and 9EH equals $(9\times16+14)$, also 158.

Working with binary and hex becomes easier as you do more of it. And learning these other number bases is necessary if you want to speak in your computer's native tongue. ■

```
00100 ;*********************************
00110 ;*                               *
00120 ;*     The Second Program --     *
00130 ;*                               *
00140 ;*     Fill the screen with 'HI' *
00150 ;*                               *
00160 ;*********************************
00170 ;
00180 VIDEO   EQU     3C00H           ;3C00H = top of screen
00190 ;
00200         ORG     7000H
00210 ;
00220 START   LD      HL,VIDEO        ;HL==> Top of screen
00230         LD      BC,400H         ;BC=length of screen
00240 LOOP1   LD      A,'H'           ;Get first character
00250         LD      (HL),A          ;Put it on screen
00260         INC     HL              ;HL==> next screen pos'n
00270         DEC     BC              ;BC=number of pos'ns left
00280         LD      A,'I'           ;Get next character
00290         LD      (HL),A          ;Put it on screen
00300         INC     HL              ;HL==> next pos'n
00310         DEC     BC              ;one less pos'n left
00320         LD      A,B             ;Get MSB of count
00330         OR      C               ;Merge with LSB of count
00340         JP      NZ,LOOP1        ;Loop er of pos'ns left
00350 LOOP2   JP      LOOP2           ;Then loop forever
00360 ;
00370         END     START
00380 ;
```

Program Listing 2. Assembly-Language Lesson 2.

Line 210 is the first line of actual instruction. It has the label Start, which the assembler sets equal to the address of the instruction (7000 hex). This instruction tells the computer to load (LD) the A register with 48 hex, the ASCII value of the character H.

Line 220 instructs the Z80 to take the current value in the A register, and place it in memory at address 3C00 hex. Remember that the screen memory occupies addresses from 3C00 hex to 3FFF hex. Therefore, this instruction causes the Z80 to place the value 48 hex in the first location of screen memory. The computer's screen electronics then print an H in the upper left-hand corner of the screen.

Four things in line 220 require special attention. First, notice the parentheses placed around the screen address. They are necessary to indicate to the assembler that 3C00 hex is an address, not a value you want manipulated. The instruction is best read as "Load the contents of register A into the memory location represented by the value 3C00 hex."

Second, notice that getting the H to the screen requires two instructions. The first instruction reads the H from memory and the second stores it to a different place in memory. In general, the Z80 cannot simply move data from one memory location to another. It only moves data to or from the registers in the chip. (Some instructions avoid this restriction, however.)

Third, it is important to realize that this instruction doesn't change the contents of register A. It copies the contents to location 3C00 hex, but A is not empty. An LD instruction never changes the value that it moves; it only transfers that value to a new place.

Fourth, notice the order of the two operands of each LD instruction. The first operand receives the value transferred from the second. If you read line 220 as "Load 3C00H from A" you can remember the correct order of operands.

Lines 230 and 240 are now clear. They load the A register with the ASCII value of I and place that value at the next screen location. When the computer executes the instruction in line 240, it prints HI in the top corner of the screen.

Line 250 introduces a new instruction, the Jump (JP) instruction. It tells the Z80 to take its next instruction from a new place in memory (like Basic's GOTO). Notice that the line gives the symbol LOOP, and the instruction says "Jump to LOOP." This instruction produces a tight, endless loop similar to the Basic instruction "50 GOTO 50." The computer seems to lock up and there's no way to regain control except by pushing the reset button. This instruction is necessary to stop the Z80 from wandering off and trying to execute whatever it finds scattered through memory. Every program must come to some specific end, either with a loop or a return to Basic or DOS.

The last line of Listing 1 uses the pseudo-op END for two purposes. First, the assembler needs to know the end point of the program. Every source code must have END in its last line. Second, if the address of the beginning of the program follows END, the assembler includes that information on disk or tape so the program runs automatically. Start is the label for the beginning of this short program and, because it already equals 7000 hex, it tells the assembler that the program starts at that address.

Before you read further, try to enter and assemble the program in Listing 1. Use the Insert command to get automatic line numbering, and copy the program exactly (you may leave out the remarks if you wish), using the right-arrow key to tab between columns on the screen. Then return to the prompt.

When you're ready to assemble the program, first try a test assembly by assembling the program with no output. The command you give to the assembler is probably A ,NO, or A/NO which asks it to assemble without output. If you enter the program correctly, the assembler displays 00000 Errors. That is the assembler's method of saying that each line has correct syntax; obviously, the assembler doesn't know whether your program does what you want it to.

Save the source code to either tape or disk. (If you make a logic error in your program, you want to correct the source code, not write it all over.) Assemble the program to tape or disk. In both cases, give the program a name—on tape the names can be anything you wish; on disk, write the source code with a /SRC or /ASM extension and the program with a /CMD extension to make it run directly. Finally, use SYSTEM or DOS to load and run your program. "HI" should appear in the upper left corner of the screen. Your computer is in an endless loop, so press the Reset button to regain control.

If all goes well, reload the editor/assembler and the source code and try to modify it—perhaps have the program print your name on the screen. Experiment with different messages and different screen locations—it is the only way to learn Assembly language.

Modifying a Program

I know Listing 1 isn't an exciting program, but your first Basic program probably wasn't much fun either. Here I add a few bells and whistles, as well as some new concepts.

Program Listing 2 fills the entire screen with the letters HI. It also introduces several new programming techniques, the first of which is in line 180. The EQU pseudo-op sets the label Video equal to the value 3C00 hex; instead of having to remember a value each time you use it in a program, you can give it an easy-to-remember label.

Line 200 looks familiar. Line 220 tells the computer to load the value of Video into the HL register pair. In this program, HL points to, or holds, the address of the current video position in order to simplify the program.

Line 230 loads the BC register pair with the value 400 hex (which equals 1,024). This program uses BC as a byte counter; 400 hex equals the length of the screen, and when BC is zero, the screen is full. Using a byte counter is similar to using a For...Next loop in Basic.

Line 240 loads the A register with the ASCII value for H. By placing the H inside a set of single quotation marks, you tell the assembler to translate the letter into its ASCII equivalent—you don't have to look it up in a table. Line 250 loads the value in the A register into the memory location to which HL points. Notice that parentheses enclose the HL

"The screen should fill with the word HI."

just like the addresses in the last program, which tells the Z80 to treat the current HL value as a memory address.

Line 260 increments (increases by 1) HL so it points at the next memory location and line 270 decrements (decreases by 1) BC so it holds the number of screen positions remaining. Lines 280–310 repeat the whole process for the letter I.

Lines 320 and 330 test the value in BC to see if it is zero yet, which means the screen is full. First, the program copies the value in the B register into the A register. Then it ORs the value in C with the current value in A. If both B and C are zero, the result of the OR is zero, and the program sets the zero flag in the F register. Otherwise, the result is some other value and the F register flag indicates Not Zero.

Line 340 uses the results of the OR to decide whether or not to repeat loop 1. It tells the Z80 to jump only if the F register shows Not Zero. If the flag indicates a zero result of the OR in the previous instruction, the program ignores the Jump command. The jump continues until the BC register pair decrements to zero. Then line 350 performs an endless-loop jump.

Notice that the instructions from lines 240–340 do not depend on any particular values in HL or BC. Whatever its current value, HL points to the current address on the screen. The program in-

```
00100 ;********************************
00110 ;*                              *
00120 ;*     The Third Program --     *
00130 ;*                              *
00140 ;*     Fill the screen with a   *
00150 ;*     simple message           *
00160 ;*                              *
00170 ;********************************
00180 ;
00190 ;
00200 VIDEO   EQU     3C00H           ;Address of screen top
00210 ;
00220         ORG     7000H           ;Beginning of program
00230 ;
00240 ;First, clear the screen
00250 ;
00260 CLEAR   LD      HL,VIDEO        ;HL==> screen top
00270         LD      BC,400H         ;BC = Length of screen
00280 CL10    LD      (HL),' '        ;ASCII Space to screen
00290         INC     HL              ;HL==> next screen pos'n
00300         DEC     BC              ;Decrease count
00310         LD      A,B             ;Get MSB of Count
00320         OR      C               ;Merge LSB of count
00330         JP      NZ,CL10         ;Loop back until BC=0
00340 ;
00350 ;Now that screen is clear, display message
00360 ;
00370 PRINT   LD      HL,VIDEO        ;HL==> screen top
00380         LD      DE,MESSAG       ;DE==> beg. of message
00390 PR10    LD      A,(DE)          ;Get next character
00400         LD      (HL),A          ;And put it on screen
00410         CALL    INCMSG          ;Increment & check DE
00420         CALL    INCVID          ;Increment & check HL
00430         JP      NZ,PR10         ;Loop until screen's full
00440 ;
00450 ;The screen is full, look for a keystroke then start over
00460 ;
00470 KEY     LD      A,(3BFFH)       ;Check the keyboard
00480         OR      A               ;Set the flags
00490         JP      Z,KEY           ;Loop until key is down
00500 ;
00510 ;Key is pressed -- wait until it is released
00520 ;
00530 NOKEY   LD      A,(3BFFH)       ;Check the keyboard
00540         OR      A               ;Set the flags
00550         JP      NZ,NOKEY        ;Wait until key is up
00560         JP      CLEAR           ;Then start again
00570 ;
00580 ;Now come the subroutines
00590 ;
00600 ;Increment the message pointer until it points to
00610 ;  the end of the message and then reset it to the
00620 ;  beginning.
00630 ;
00640 INCMSG  INC     DE              ;DE==> Next pos'n of msg.
00650         LD      A,(DE)          ;Get the character
00660         CP      0               ;Is it a 0 ?
00670         RET     NZ              ;Return if not
00680         LD      DE,MESSAG       ;Else DE==> beg. of msg.
00690         RET                     ;And then return
00700 ;
00710 ;Increment the video pointer until it is off the screen
00720 ;
00730 INCVID  INC     HL              ;HL==> Next pos'n of vid.
00740         LD      A,H             ;Get MSB of pointer
00750         CP      40H             ;If H=40H, off the screen
00760         RET                     ;Return for test
00770 ;
00780 ;Now type in the message
00790 ;  (You may use any message you wish)
00800 ;
00810 MESSAG  DEFM    'How is this for fast? '
00820         DEFB    0               ;Mark end of message
00830 ;
00840 ;That's all, so end the program
00850 ;
00860         END     CLEAR           ;Include starting address
00870 ;
```

Program Listing 3. Assembly-Language Lesson 3.

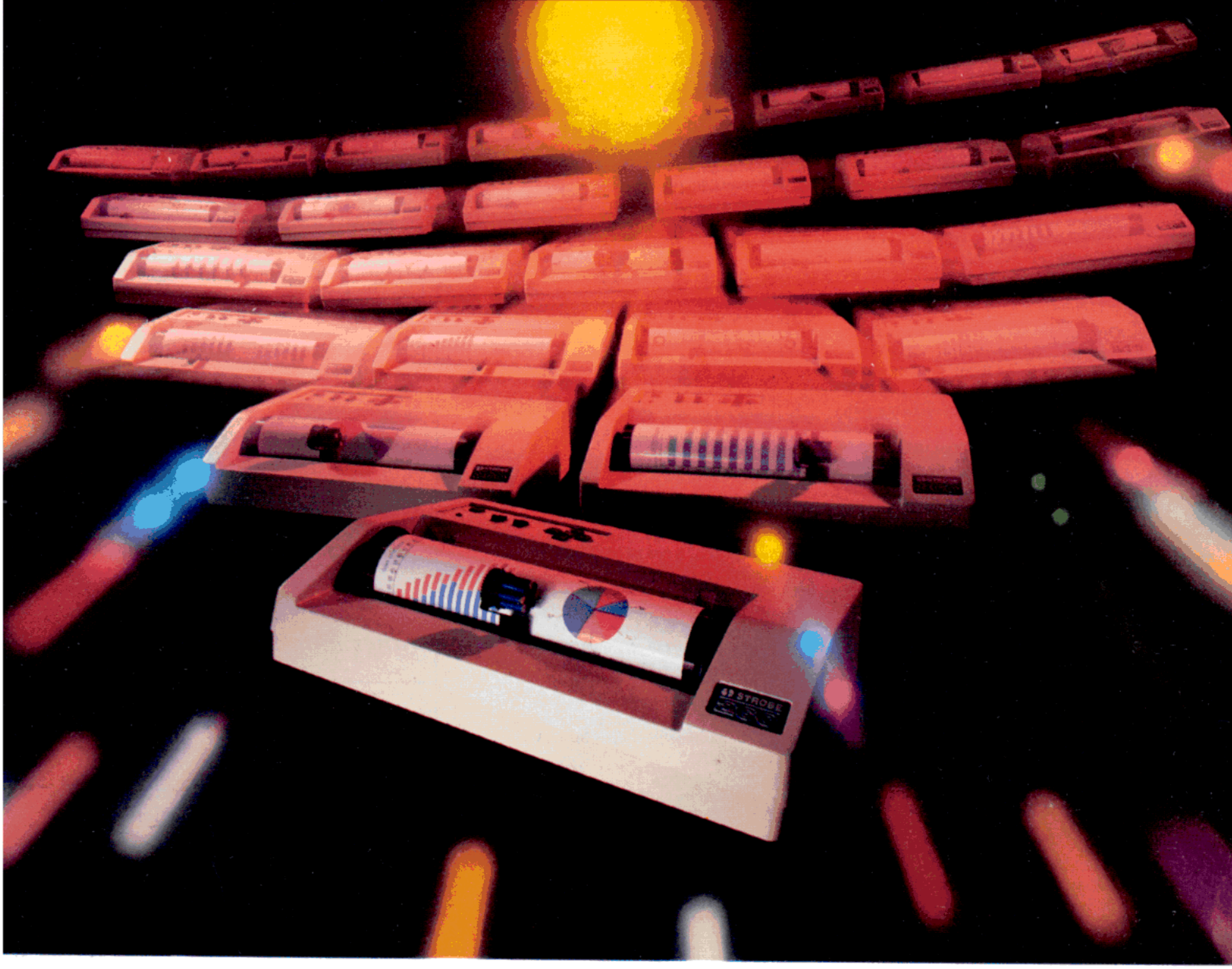
crements it after each character prints. BC's current value decrements after the program prints each character. The program continues until BC indicates a full screen. The entire routine depends on the Z80's ability to make a conditional jump in line 340, based on the current status of the zero flag.

Enter, assemble, and run Program Listing 2. Again, you must use the reset button to exit from the program because of the endless loop in line 350. The screen should fill with the word HI. See if you can modify this program being careful that you don't let the program print off the end of the screen.

More Bells and Whistles

This next program lets you fill the screen with any message you wish. It also allows you to hit any key to clear the screen and print the message again.

Lines 190–270 in Program Listing 3 need no explanation. Notice that the instruction in line 280 loads a character (here, an ASCII space) directly into the memory location to which HL points. This instruction avoids first loading the



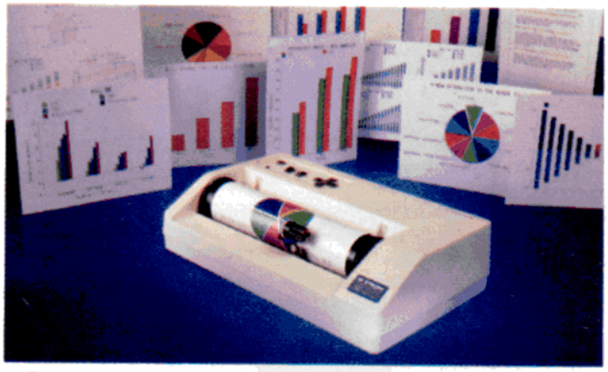
The Best Just Got Better

Introducing the new Strobe Models 260 and 200 Graphics Plotters. How do you make the world's most cost-effective graphics system even more remarkable? Simple. Make it easier to use, more powerful, more versatile and more attractive. The hard part is doing all of the above while holding the line on a very affordable price. With the new Models 260 and 200 Graphics Systems, Strobe has done it all. And the benefits are all yours.

The Strobe 260 is the first eight-pen plotter and software package available for under $1,000. With just a few keystrokes on your computer, the system produces superb high-resolution charts, drawings and transparencies with up to eight automated color breaks. The even more economical Model 200 produces the same wide range of graphics with a single pen which may be changed manually.

The new Strobe plotters interface with virtually every popular microcomputer made today, including the Apple II, IIe and III, Victor, IBM PC, Kaypro and Osborne systems. What's more, the StrobeWare software programs provide every graphic format you will probably ever need, including multicolor pie charts, bar graphs, line graphs, text processing, and a broad selection of symbols and character styles. To further increase your capabilities, the new Strobe systems also interface with over 40 other popular graphics programs.

Every feature of the new Strobe Graphics System is designed to give you the highest quality graphics quickly, easily and inexpensively. The system also looks very nice on your desk. So visit your local dealer today and see how much better the best can be.



Apple II, IIe and III, Victor, IBM PC, Kaypro and Osborne are trademarks of Apple Computer, Inc., Victor United, Inc., International Business Machines Corporation, Kaypro Corporation, and Osborne Computer Corporation, respectively.

STROBE INCORPORATED

Strobe Inc.
897 5-A Independence Ave.
Mountain View, CA 94043
415/969-5130

The Strobe Graphics System

Seeing is believing

338

character into the A register and then back into a new location in memory.

Lines 290–330 are similar to the last program. They increment HL, decrement BC, and test to see if BC equals zero. The entire block of code from lines 260–330 clears the screen by filling it with spaces.

The second block of the program, lines 370–430, prints a message repeatedly until it fills the screen. HL again points to the screen location. Now, however, DE points to the message in line 810. These lines load 1 byte of the message into the A register in line 390, then put that character into screen memory in line 400.

> *"Keep experimenting and you will soon find that Assembly language is as easy as any other computer language."*

Lines 410 and 420 introduce a new mnemonic, Call. This instruction is similar to Basic's GOSUB command in that it runs a subroutine. The Z80 saves the address of the next instruction before it runs a subroutine in order to know where to return when it completes the subroutine. Then it jumps to the subroutine and performs those instructions.

The label INCMSG, which stands for "Increment Message Pointer," defines the first subroutine. The subroutine starts at line 640. This line increments DE, which points to the message. Once incremented, line 650 loads the new character to which it points into A. The command CP zero in line 660 means "Compare the value in A with zero. If the two values are identical, set the flag to Z to indicate a true compare; otherwise set the flag to NZ."

The RET command in line 670 means "Return from this subroutine." Because NZ follows it, it is a conditional return—the computer executes the return only if the flag in the F register indicates Not Zero. Otherwise, in line 680, the program points DE to the beginning of the message again, and then an unconditional return in line 690 sends the Z80 back to the instruction in line 420. The entire subroutine INCMSG depends on a zero byte at the end

Continued on p. 86

Beginner's Guide to Assembly-Language Terminology

Simple Glossary

This short glossary should help you learn the most common Assembly-language terms. It's arranged in logical, rather than alphabetical, order.

bit (binary digit)—Though sometimes defined in terms of electronic signals, for present purposes a bit is the smallest amount of information on which the computer acts. It is a single 1 or zero digit and the building block for all computer notation.

byte—A sequence of 8 bits. A byte is generally thought of in terms of its total value. With 8 bits you can express any value between 00000000B and 11111111B. In normal decimal notation that's any value between 0 and 255; in hexadecimal it's any value from 00 hex to 0FF hex.

word—In computer jargon, a word comprises 2 consecutive bytes, or 16 bits, and the computer handles it as a single value. The value of a word can vary from 0 to 65535 (decimal), from 0000 hex to FFFF hex, from 00000000 00000000B to 11111111 11111111B in binary.

address—A one-word (2-byte) value that denotes any byte of memory. Every byte of memory has a unique address, allowing you to find it easily, and find and manipulate its value.

LSB and **MSB** (least significant byte and most significant byte)—Since every word and every address is composed of 2 bytes, it is useful to have a name for each. The most significant byte is the high-order byte—the one that normally comes first. The other is the least significant byte. For example, in the hexadecimal word 3C42 hex, 3C hex is the most significant byte and 42 hex is the least significant byte. Because of the way the internal Z80 circuitry works, it usually stores 2-byte values in memory backwards, with the LSB before the MSB.

ASCII code—Every possible video character, all video control characters, all graphics characters, and all special (Model III) characters have a unique 1-byte code that the video circuitry uses. The codes for letters and numbers are standard between various computers, but graphics and special characters are not. The ASCII code for the character 1 is 49 or 31 hex or 00110001B (rather than the seemingly obvious value of 1, a complicating fact of life for machine-language programmers).

register and **register pair**—Registers are special memory inside the Z80 chip. Each is either 1 or 2 bytes in length (8 or 16 bits). A register pair is two 8-bit registers that can work together as a single 16-bit register. There are 22 registers inside the Z80 that your programs can manipulate directly.

ROM (read-only memory)—ROM is unalterable, unchangeable memory inside the computer that holds the resident Level I or Level II Basic inside the Models I and III. This memory doesn't change even when you shut off the computer's power, so your computer never "forgets" its knowledge of Basic.

RAM (random-access memory)—Sometimes called read-and-write memory, or program memory, this memory is changeable. The computer uses it to store both programs and data. Unfortunately, it loses all of the information stored in it if the power to your computer is turned off. Unless you did some unusual home-brew modifications, your computer has 4K, 16K, 32K, or 48K bytes of RAM. (K stands for 1000, but in computer usage usually means 1024. 16K, therefore, means 16384 bytes of RAM.)

Status Flags

The F, or flag, register contains 4 commonly used status bits, each of which can direct conditional jumps, calls, and returns.

Z and **NZ**—The zero flag is the most common. This bit indicates whether the result of a previous operation was zero or some other value. For example, if the result of a subtract or compare operation is zero, the flag shows zero. In source code, the condition Z means zero and NZ means not zero. The zero

Continued on p. 86

Continued from p. 84

of the message to mark its conclusion.

Line 420 sends the Z80 out to another subroutine, INCVID (Increment the Video Pointer), and passes control to the instructions starting in line 730. First line 730 increments HL. Remember that the video memory is in address from 3C00 hex to 3FFF hex. If HL equals any value in that range, it points at the screen and everything is okay. But when HL equals 4000 hex, it points to the first memory address past the screen. Then H holds the value 40 hex and L holds the value 00 hex.

Line 740 loads the new value in the HL register into the A register for testing. Line 750 compares that value to 40 hex in order to set the F register flags. You might expect the next instruction to be a conditional jump or return, but it isn't. Instead, the subroutine ends with an unconditional return (RET) command. However, the return command doesn't affect the F register, and so the flags remain the same when the Z80 returns to the instruction in line 430, which does contain a conditional instruction. Very often in Assembly language, programs set the condition flags with one instruction but don't use them until several instructions later. Be sure that the instructions between don't change any of the flags.

Line 430 repeats the print block until the screen fills. Then the program continues with the KEY block starting with line 470. Here, the value in memory location 3BFF hex loads into the A register, which is part of the keyboard memory. This location holds a zero if you're not pressing any key; if it holds any other value, you're holding down at least one key.

Unfortunately, loading a value into the A register doesn't set the condition flags. Therefore, line 480 ORs register A with itself. The value in A doesn't change, but the flags are set to indicate whether its current value is zero. If it is, the conditional jump in line 490 sends the Z80 back to KEY to wait for you to press a key.

After you do so, the program enters the final block of the program, NOKEY. Another loop waits for you to release the key. Then, and only then, the program loops back to the Clear routine at the beginning, clears the screen, and starts all over.

Lines 810 and 820 demonstrate two new pseudo-ops. In line 810, DEFM means "Store the text between the single quotation marks in memory here." It loads whatever you type in into memory and sets the label MESSAG equal to the address of the first character.

Finally, in line 820, DEFB means "store 1 byte of this value into memory." This instruction stores a byte

Continued from p. 85

flag generally checks the results of arithmetic and logical operations, and tests whether a single register decrement (DEC) results in zero.

C and **NC**—The carry flag indicates whether a carry or borrow occurred that gives an arithmetically inaccurate answer. For example, if a program adds two 8-bit numbers and produces a 9-bit result, the carry flag holds the 9th bit. Also, rotate and shift instructions use the carry flag extensively.

P and **M**—The normal arithmetic convention for working with signed numbers uses the highest-order bit to indicate the sign. Therefore 1 indicates a negative number and zero indicates a positive number. The sign flag shows whether the result of a signed arithmetic operation is plus (P) or minus (M). This flag is also of use with compare operations.

PE and **PO**—The last testable flag has two functions. As a parity flag, it indicates whether a byte has an even or odd number of bits equal to 1. PE then means even parity and PO means odd parity. The same set of flags can also indicate whether there is an overflow from an 8-bit or 16-bit addition or subtraction, in which case PO means no overflow and PE indicates that an overflow has occurred.

N and **H**—There are two final flags in the Z80, N and H, which you cannot test directly by a program. The computer uses them in a special type of arithmetic called "binary-coded decimal" and generally accesses them by the special DAA instruction.

Logical Operations

The Z80 performs three kinds of logical operations: OR, AND, and XOR. Each has specific uses in Assembly-language programs.

Each logical command works on a single bit position at a time. For ex-of zero into memory to mark the end of the message. This is the zero byte that the subroutine INCMSG tests for. Never assume that any memory location holds some specific value. If your program looks for a byte of zero at the end of a message, you must put that byte there.

Be sure to try a test assembly of Listing 3 before you save the source code and assemble the program. With a program of this length, you can easily make a typing mistake that you must correct before the program runs correctly.

Once you get this program running, it demonstrates the speed of machine language. When you press and release a key, you see a very brief blink on the screen. In that time, it prints a space in every screen location and prints the message again until the screen is full. Even though it must print 2,028 characters, it all happens in a flicker of the screen.

Learning More

No one can learn any computer language from one article. So far you can use only a few mnemonics and a few of the many types of program logic. You are on your way, but still have much to learn.

You can do many things to develop your Assembly-language programming skills. First, read Assembly-language programs in magazines carefully and try to follow the logic of each. Most magazine programs have a lot of remarks to help you understand what they do.

Second, read through the list of Z80 mnemonics (there is an excellent list with complete explanations in the Series 1 manual) and try to imagine how you can use each variation of each instruction. Also, you might read one of the books available on beginning Assembly-language programming. William Barden's *TRS-80 Assembly-Language Programming*, available from Radio Shack, is one of the easiest to read.

Third, if you have some short machine-language programs available in your library, try to disassemble one and understand what it does without the benefit of remarks. Don't try to disassemble something as long as Scripsit or VisiCalc; such complexity will completely overwhelm you.

Most important, write your own Assembly-language programs. The more you write, the easier it becomes. Keep trying, keep writing, keep experimenting and you will soon find that Assembly language is as easy as any other computer language. ■

Write to Hardin Brothers at 280 North Campus Ave., Upland, CA 91786, or contact him through CompuServe. His e-mail address is 72165,735.

ample, when the Z80 uses OR to combine two values, first it ORs their highest bits, then their next highest bits, and so on. Logical operations have no concern with the total values of the operands; bit patterns are the only concern.

AND—When the Z80 ANDs, the result bit is a 1 only if both of the operand bits are 1's.

```
    10010011
AND 10100110
    10000010
```

Use AND to mask out unwanted bits. For example, if you want to isolate just the 4 lowest bits of an 8-bit byte, you can AND that byte with 00001111. The result copies the 4 lowest bits of the original byte into the result and sets the 4 highest bits to zero.

OR—The result bit of 2 ORed bits is a 1 if either or both of the operand bits are 1's.

```
   10010011
OR 10100110
   10110111
```

OR merges two values together and forms bit records in which each bit in a byte has an individual meaning.

XOR (Exclusive OR)—The result bit of 2 XORed bits is a 1 if either, but not both, of the operand bits is a 1.

```
    10010011
XOR 10100110
    00110101
```

Programs use XOR less often than OR and AND. Its major use is in clearing the A register and flags with the command XOR A.

An interesting fact about the XOR operation is that it is cyclic. For example, if you XOR A and B, and then XOR the result by B again, the second result is the original A value. ■

✓ See List of Advertisers on Page 291

LOAD 80

Assembly-Language Disk I/O

by David G. Haan

With a knowledge of Assembly-language disk input/output, you can make your programs run more efficiently and solve most I/O problems.

Assembly-language disk input/output (I/O) is a complicated and advanced technique, but understanding it allows you more creativity in programming and more knowledge in troubleshooting and repairing disk I/O problems.

This article deals with the use of the Model III TRSDOS disk I/O and explains the subroutines used in reading and writing disk files. Included is a demonstration program, TESTFILE/TST, that chains these ROM routines and shows how they manipulate disk files.

Within the text, I explain how to use each of the routines and what you can expect on entry and exit. I also point out some errors in Radio Shack's explanation of these routines. You should have a copy of the *Disk System Owner's Manual* for reference.

In order to see the results of each routine, I provide a non-destructive exit routine so you can examine the areas in memory that are affected. I list the routines alphabetically in the first section of EQUs.

I don't intend here to teach coding practice, but merely to demonstrate steps necessary to use each of the routines. At the end of this article, you should have sufficient knowledge to write Assembly-language disk I/O routines.

I will use the Program Listing (TESTFILE/TST) as a demonstration program throughout this article. If a file by the name of TESTFILE/TST exists on the disk you want to use, you should rename it.

Table 1 lists the major storage areas and buffers TESTFILE uses for system routines and their functions.

The program sets up registers before each call to a system routine. This is not always necessary since some of the registers used remain unchanged when returning from a previous routine. It is done, however, to indicate the information required prior to calling a system routine.

Following along with the Listing helps you see what registers the routines need, and what they need to contain prior to executing the routines. Table 2 shows the condition of the registers prior to and after each routine's execution.

Storage Areas	Bytes Used	Description
BUFFER	256	Operating system uses this as a data storage area during a disk read or write. It must be 256 bytes long since all reads and writes are done one sector at a time.
UREC	less than 256	A user buffer that locates all the data you want to write to or read from disk. The system moves data between UREC and Buffer if the logical record length is less than 256 bytes.
DCB	64	Data control block contains information used by the system to read or write data from or to the disk. For a layout of the DCB, see Table 3 or the *Disk System Owner's Manual*.
KBLNBF	16	Buffer used to hold data entered via keyboard. Used in conjunction with system routine KBLINE.
DIRBUF	24	This is where RAMDIR places a single directory entry. The format of this buffer is shown in Table 4 or under RAMDIR in the *Disk System Owner's Manual*.

Table 1. Storage areas and buffers.

Program Operation

The demonstration program starts by saving the HL register (see the Listing). The HL register contains the address of the first non-blank character following the last command you entered under

The Key Box

Model III
32K RAM
Assembly Language
TRSDOS, Editor/Assembler, Disk Drive


Datalife
Verbatim
Minidisk
FLEXIBLE DISK
CONTROL DATA
maxell
Mini-Floppy Disk
MD1
CERTIFIED and TESTED
CONTENTS: 10

TRSDOS READY. Keep this fact in mind for later reference.

Then the program checks to see if this is your initial entry into this program. It does so by looking at a storage location called Entry that contains the address of the location at which you reenter the program (if you have reentered it).

If the reentry address is between 8000 hexadecimal (hex) and 8FFF hex, Entry uses it as the reentry address. The program places the contents of Entry there with a routine called Look. Look is the non-destructive exit routine the program calls each time you execute a system program and provides a place in memory where you can look to see the effects of the routine's execution.

When you run this program for the first time, it clears the storage areas from Buffer to LKBUF. This makes it easier to see what data the system routines put into memory. Also, the program puts dummy data into the area called UREC. The program then writes this data to disk. Following this, a message indicates the start of the program.

The program then prompts you for the name of a file to which you want to write data. This is where you type in the filespec you want used in the disk's data control block (DCB). The code that asks for the filespec starts at the label GETSPC. With the B register set to 15, and the HL register set to the keyboard input buffer (KBLNBF), the system routine KBLINE lets you enter up to 15 characters for a file name.

Once you do so and hit the enter key, the system's Syntax routine checks to see if your filespec has the correct syntax. If it does, the subroutine duplicates the filespec in the DCB. Syntax uses the HL register, which points to the filespec in the keyboard input buffer (KBLNBF), and the DE register, which points to the DCB.

On return, Syntax sets or clears the zero flag, depending on whether or not the filespec syntax was good. If it is, the subroutine sets the zero flag. If not, it resets the zero flag and the program prompts you for the filespec again.

To check this system, enter any file name preceded by a blank such as BADSPEC. Press the enter key. Since this enters a bad filespec (the first character is a blank), the program displays a message asking for a filespec again.

You might think that since the file name must begin with an alpha character and have eight or fewer characters, Syntax would check this. It doesn't. It does check to see that the filespec starts with a blank, but Syntax allows the file name to start with a number and simply truncates it to eight characters.

Also, if you use an extension and password, Syntax truncates the extension to three characters, the password to eight characters, and doesn't return an error. If you need to check for the accuracy of your filespec, better do your own checking rather than rely on the Syntax routine.

Now enter TESTFILE as a file name and press the enter key. Since this file name has the correct syntax, the subroutine copies it to the DCB and the local routine Look displays a message asking you if you want to look at the results. If you answer with anything other than Y, the program continues execution, so answer with a Y. This brings you into Debug.

You can go directly to Debug from the demonstration program by using the system routine called COMDOS. COMDOS executes any command you can execute while in TRSDOS READY from a user program. The HL register must point to the address which has the command you wish to execute. The program then executes a jump to COMDOS.

The Look routine points the HL register to the label EXECUT in the message area of the listing, which contains the program name Debug followed by a carriage return. Once in Debug, you can use the command D to look at both KBLNBF and the DCB.

To view KBLNBF, enter its address found in the section of equates at the beginning of the program under Storage Locations. In KBLNBF, note the file name TESTFILE followed by 0DH, a carriage return. Now look at the DCB. You can find this address listed in the section of equates also.

Notice that the file names in KBLNBF, which is now below the DCB, and the DCB are the same except for one thing. The file name TESTFILE in KBLNBF is followed by 0DH while the file name in the DCB is followed by 03H. The TRSDOS *Disk System Owner's Manual* says the file name in the DCB is followed by 0DH. As you can see, this isn't true if you use the Syntax routine.

To return to the demonstration program, press the Q key to exit Debug and type in the name under which you assembled the demonstration program. This returns you to where you left off. The program displays a message on the screen indicating the location at which you reentered the program. In fact, each time you reenter the program, a message indicates the reentry point.

You should reenter the demonstration program at the label Extend. You can now enter a three-character extension that the program adds to your file name in the DCB via the PUTEXT routine. Of course, you could have added the extension to the file name when you originally entered the filespec, but the

	Registers					
Routine	**AF**	**BC**	**DE**	**HL**	**IX**	**IY**
BKSPC	Yes	No	No	No	No	No
CLOSE	Yes	No	No	No	No	No
CMDDOS	N/A	N/A	N/A	N/A	N/A	N/A
CMDTXT	N/A	N/A	N/A	N/A	N/A	N/A
COMDOS	N/A	N/A	N/A	N/A	N/A	N/A
DIVIDE	Yes	No	No	Yes	No	No
DSPDIR	Yes	Yes	No	Yes	No	No
ERRDSP	Yes*	No	No	No	No	No
FILPTR	Yes	Yes	No	No	No	No
INIT	Yes	No	No	No	No	No
JP2DOS	N/A	N/A	N/A	N/A	N/A	N/A
KILL	No	No	No	No	No	No
OPEN	Yes	No	No	No	No	No
POSEOF	Yes	No	No	No	No	No
POSN	Yes	No	No	No	No	No
PUTEXT	Yes	Yes	No	Yes	No	No
RAMDIR	No	No	No	No	Yes	No
READ	Yes	No	No	No	No	No
REWIND	Yes	No	No	No	No	No
SYNTAX	Yes	Yes	No	Yes	No	No
VERF	Yes	No	No	No	No	No
WRITE	Yes	No	No	No	No	No

Yes = registers have changed. No = registers have not changed. N/A = not applicable.

Only the primary registers change; the demonstration program doesn't use the alternate set.

* Only the flag register changes.

Table 2. Register conditions after routine execution.

PRICES AND SERVICE TOO GOOD TO PASS UP!

PRODUCTS FOR THE TRS-80®

SOFTWARE

NEWDOS80/Vers. 2	$124.95
COMREF for Newdos80	13.95
LDOS 5.1	109.95
DOSPLUS 3.5	119.95
DOSPLUS IV	119.95
6.0 Plus	39.95
DOSPLUS II	195.00
Micro Clinic - Mod 1	24.50
Micro Clinic - Mod 3	28.50
Super Utility + 3	59.95
Trashman	34.50
Faster	24.95
R.P.M. - Disk Timer	22.50
Basic Editor by C.A.U.	24.75
M-ZAL Release Two by C.A.U.	119.50
LDOS Utility Disk #1	42.50
The Toolbox for LDOS	59.95
EDAS IV	84.95
The BASIC Answer	59.95
Dot Writer 3.0	66.50
Dot Writer 3.0 w/letterset	84.50
Dot Writer Font Disks	23.50
GEAP	38.50
Maxi Manager	119.50
Maxi Mail (Mod 3 Only)	64.95
Maxi Stat	164.95
Maxi CRAS	79.95
Powermail Plus	129.00
The Home Accountant	62.95
Tallymaster	62.95
Microterm Mod 1, 3, or 4	64.95
Omniterm Mod 1/3	79.95
Omniterm Mod II	139.95
LYNX Modem	209.00
LNW Doubler 5/8 w Dos +	194.95
LNW Doubler 5/8 w/o Dos	159.95
LNW System Expansion II	329.95
SBSG Accounting Modules Model 1 or 3	175.00
Model 2	250.00
Verbatim Disk 5″ SS/DD	26.50

WORD PROCESSING

Newscript 7.1	99.95
Newscript 7.1 plus Labels	114.95
Newscript DW Proportional	39.95
Newscript File Converter	19.95

Electric Webster + Corrections	119.50
Elec. Webster Grammar	39.95
Elec. Webster Hyphenation	39.95
Lazy Writer 3.4	139.95
LeScript Mod	109.95
Zorloff II	58.95

GAMES

	tape	disk
Attack Force or Cosmic Fighter	12.75	15.95
Stellar Escort or Galaxy Invasion	12.75	15.95
Meteor Mission II or Robot Attack	12.75	15.95
Defense Command or Super Nova	12.75	15.95
Weerd		15.95
Forbidden City or Forbidden Planet		29.95
Hyperlight Patrol		15.50
Defiance (Disk Only) or Panik	15.50	19.50
Devil's Tower	12.75	15.95
Alien Defense or Bounceoids	13.95	17.25
Sneak Thief	17.25	18.95
Frogger	17.25	19.95
Reign of Red Dragon or Stratos	19.95	19.95
Double Feature		31.95
Caterpillar or Scarfman	13.95	17.25
Armoured Patrol or Eliminator	19.95	19.95
Rear Guard or Sea Dragon	19.95	19.95
Adv. Tripac 1-3, 4-6, 7-9, or 10-12		31.75
Adventure International Hint Book		6.95

WRITE FOR OUR FREE CATALOG

6.0 PLUS

Enables the novice and expert alike to increase their programming speed and ability above the limits created by TRSDOS 6.0. 6.0 Plus provides features too numerous to mention including a utility to insert keyword spaces in Model 3 programs.

$39.50

DOSPLUS IV

Experience the full power of your Model IV with DISPLUS IV. 100% program and media compatibility with TRSDOS 6 eliminates conversion problems.

$119.50

LNW80 II COMPUTER

Includes Dosplus 3.5, CP/M 2.2, Microterm, Electric Pencil, Chart-Ex, LNWBasic, and a full accounting system including general ledger, accounts receivable, accounts payable, and payroll.

$1,599.00

- 24-Hour Shipping for Items in Stock
- Free Use of Credit Cards
- Large Selection-Call for Items Not Listed
- Toll Free Order Line
- Free Shipping on Orders over $100
- Friendly, Honest, Reliable Service

We accept Visa, Master Card, check, cash, money orders, and COD. In the 48 continental States add $2.50 for UPS standard shipping, we'll pay shipping if your order is over $100. Alaska and Hawaii orders are charged actual shipping charges. COD orders are charged an extra $2.00 and require cash or certified check on delivery. POs accepted upon approval.

When ordering by mail, include your telephone number, credit card information, computer model, memory size, and number of drives. Colorado residents add appropriate sales tax (6½% in Denver).

Prices are subject to change without notice.

Applied Microsystems, Inc.
612 Washington, Denver, CO 80203 ✓390

ORDER NOW TOLL FREE **1-800-468-4474**
IN COLORADO CALL 861-9250

extension is being requested here to demonstrate the routine. Enter TST as the extension.

To work properly, PUTEXT requires that the HL register point to the location of the extension and the DE register point to the DCB. The demo program then calls PUTEXT. There are no exit conditions to check for, so the program calls Look so you can again check the results in KBLNBF and the DCB. In the DCB, the filename now appears as TESTFILE/TST followed by 03H. Now return to the demonstration program.

You should have reentered at the label OPNFIL. Here the program attempts to open the file using the filename TESTFILE/TST. If the file had already existed, it would simply have been opened, but since the file doesn't exist, the program displays the message File Not Found. You can look at the DCB and see that nothing happened.

Before going on, look at the local routine ERPROC which contains the system routine called ERRDSP. You can find this in the Listing under the banner Error Display Processor. ERRDSP is a system routine that displays an error code or expanded error message on the CRT. It then either returns to the user program, or to TRSDOS READY. Setting bit 6 masks the error code that is in the A register. The routine spells out the full error message.

If bit 7 is set, control returns to the user program. How you process an error is up to you. You can abort the program, try again, skip processing, or whatever you feel is appropriate. Now return to the demonstration program.

You should reenter the program at label NEWFIL. This initializes a new file and you can again look at the DCB. The parameters that the HL, BC, and DE registers pass are the same whether you open or initialize a file.

In this case, the HL register contains the address of a 256-byte buffer called Buffer where the operating system does the actual writing to and reading from the disk. The DE register must point to the DCB, and the B register must contain the logical record length.

A physical record is 256 bytes long or one disk sector. You can subdivide the physical record into smaller segments called logical records. A logical record is a record of from 1 to 256 bytes long. In the EQUs, I define the logical record length (LRL) as 128 bytes long.

If you define a logical record as 64 bytes long, four logical records comprise one physical record ($4 \times 64 = 256$). However, if a logical record does not evenly divide into a 256-byte physical record (such as a logical record of 50 bytes), the system spans physical records to keep your logical records intact.

Byte	Contents
0–2	Reserved for system use.
3–4	Address of system I/O buffer. (BUFFER)
5	Offset into the current physical record of the end of the last logical record read or written.
6	Drive number on which the file exists.
7	Reserved for system use.
8	End Of File offset to the end of the last logical record.
9	Logical record length.
10–11	The next physical record where a read or write will take place.
12–13	The last physical record in the file.
14–63	Reserved for system use.

Table 3. Data control block layout.

Byte	Contents
0–14	File name/ext:d left justified and padded with spaces.
15	Protection level of file. 0 to 6.
16	End Of File offset to the end of the last logical record in the last physical record.
17	The logical record length.
18–19	The last physical record number in the file. LSB is in byte 18 and the MSB is in byte 19.
20–21	The number of granules allocated to the file. LSB is in byte 20 and the MSB is in byte 21.
22–23	Two plus marks indicating the end of the directory.

Table 4. Directory layout in RAM.

It's possible to have a logical record equal in size to the physical record (256 bytes), but this requires additional programming. I'll discuss how the data moves to and from disk and the special requirements for 256-byte logical records later.

For now, answer Y to the question on the screen and take a look at the DCB. Notice that the program file name in the DCB no longer appears. Instead, the DCB's layout is as shown in Table 3. Bytes 3 and 4 of the DCB point to the 256-byte buffer called Buffer. Remember, the low order byte appears first, the high order byte second.

Byte 5 represents the offset to the delimiter at the end of the current physical record. Byte 6 indicates the drive number on which the file resides. Byte 8 is the end-of-file offset of the last delimiter in the last physical record. Byte 9 shows the logical record length, and bytes 10 and 11 display the next physical record number with bytes 12 and 13 giving the ending physical record of the file.

Figures 1 and 2 show examples of how to interpret the contents of bytes 5, 8, 10, and 11 of the DCB. Figure 1 shows two physical records, or sectors, of 256 bytes each, comprising four 128-byte logical records.

Suppose you just read the first logical record (record zero) of physical record zero. Byte 5 of the DCB now contains 80 hex and bytes 10 and 11 of the DCB contain 0000 hex. The DCB now points to the first byte of the next logical record.

Logical record zero goes from bytes 0 hex to 7F hex, and logical record 1 goes from bytes 80 hex to 0FF hex. Byte 5 of the DCB actually contains the first byte following the end of the last logical record read or, for that matter, the last one written.

Figure 2 is similar to Fig. 1 (now in the second logical record of physical record zero). Here, byte 5 of the DCB contains 0 hex, and bytes 10 and 11 of the DCB contain 0001 hex. The DCB now points to the first byte of logical record 2, the first logical record of physical record 1.

Byte 8 of the DCB is similar to byte 5 of the DCB, but only applies to the last physical record in the file. It points to the first byte following the last logical record in the last physical record.

In Fig. 2, if the last logical record in the file is logical record 2, byte 8 of the DCB contains 80 hex. This is actually the first byte of the next logical record, if it existed. You should note that the value in byte 8 need not always point to

✓ See List of Advertisers on Page 291

the beginning of the next logical record. *It can point* anywhere within your last logical record. It is meant to point to the first byte following the actual end of data.

To make sure you put the correct end-of-file (EOF) offset in the DCB, you must calculate and put the actual EOF offset into byte 8 of the DCB. One way you can experiment with this is to use this program with different logical record lengths and modify the end-of-file offset in byte 8 of the DCB just before closing the file. To keep it simple, don't do it here. Now return to the demonstration program.

Write Operation

Once you return from looking at the DCB, execute the code at the WRTFIL label. The system needs two pieces of data to write a record to disk. The HL register must point to the buffer that contains the data you want to save, and the DE register must point to the DCB. The program sets the HL register to point to UREC, which contains some sample data, while the DE register points to the DCB.

When the program invokes the Write subroutine, it moves 128 bytes (the logical record length) from UREC to the area called Buffer. If the write is successful, it sets the zero flag. If an error occurs, the subroutine resets the zero flag and returns an error code in the A register for display via ERPROC.

As long as the logical record length is fewer than 256 bytes long, the system moves data from the user buffer UREC to the I/O buffer called Buffer and vice versa.

If you set the logical record length to 256 by opening a file with the B register equal to zero, you must move the data between UREC and Buffer yourself. This means that for each Write command, you must move the 256 bytes of data you want written to Buffer before the program calls the Write routine. Also, after each read, you must move the 256 bytes from Buffer to wherever you want it to go as its final destination. When the logical record length is 256 bytes, Read and Write commands ignore the HL register.

Once you complete the first write of the logical record of 128 bytes, answer Y to the question on the screen and look at Buffer and the DCB. If you look at the DCB, you see that the next physical record number at bytes 10 and 11 is zero. Also, the offset to the delimiter at the end of the current physical record at byte 5 is 80 hex, indicating that the program successfully completed one write. Records start at number zero with two 80 hex logical records per physical record. Looking at Buffer, you can see the dummy data.

Return to the demonstration program to continue with the next write at label WRT002. The program now initiates a second disk write, but here the HL register points one logical record length (1 LRL) into UREC. This occurs so the system can pick up the next block of data.

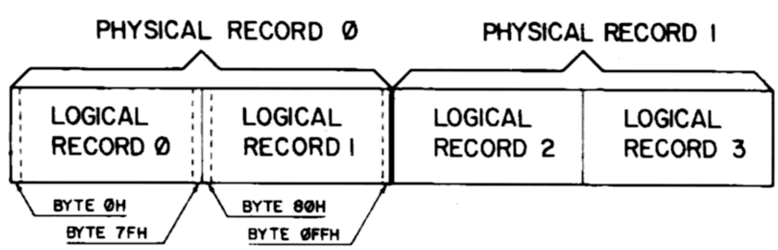


Figure 1. Byte interprets.

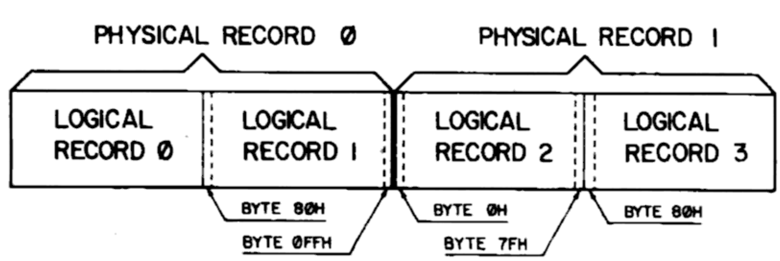


Figure 2. Byte interprets.

Make sure the HL register points to the right area of memory for the data you want transferred. If the write is successful, the program transfers a sector of data, or 2 LRLs, from Buffer to the disk. Again, you can look at the DCB and buffer areas for verification.

Now the DCB shows that the next physical record number at bytes 10 and 11 is 1 and the ending physical record at bytes 12 and 13 is also 1. The offset to the end of the current record at byte 5 is zero, indicating that the program has written two logical records.

After returning to the demonstration program at label WRT003, the program completes a third write, after which you may look at the DCB. Notice that the last half of the buffer contains junk. The program clears out Buffer after it writes a sector and loads it with data of its own choosing from somewhere in the system.

Now, return to the demonstration program at label WRT004 for a fourth disk write. Following this, at label WRT005, the program makes a fifth write. After each of these writes you may look at the DCB.

Since the fifth write is only half of a new third sector, the DCB shows that the next physical record is 2 (the third physical record), the ending physical record is 2, and the offset is 80 hex at byte 5, for a total of five 80 hex-byte logical records.

In the *Disk System Owner's Manual* the explanation of the next record number (NRN) as well as those of Read and Write operations, say that after each Read or Write the NRN increments by one. This is true only if the logical record length is 256 bytes long. If the logical record length is 128 bytes as in this case, the NRN increments only after two reads or writes.

You should now return to the program to close the file. You enter at label CLOSFL. To close a file, you need to pass only one parameter to the routine called Close. This is the address of the DCB in the DE register. When you close the file, the program makes a final write to the disk, keeps track of the end of the last logical record, and updates the DCB and the disk directory.

If you close the file successfully, the program sets the zero flag. If not, you need to process the error. Get into Debug and look at the DCB. As you can see, the file name, including the number of the disk drive where the file is located, again appears at the beginning of the DCB. If you call the directory, notice that the file has an LRL of 128 bytes, five logical records, three physi-

cal records (one granule), and an EOF of 128 bytes.

If you want to quit and take a break, go ahead, but if you turn off your computer or run some other program, make sure you reenter the program at the correct point. To do this, use the M command under Debug and modify the location Entry so that it points to GTSPC1 in the demonstration program.

Remember, the low order byte of the address of GTSPC1 goes into the low order byte of Entry, and the high order byte of the address of GTSPC1 goes into the high order byte of Entry. Now you can exit Debug and continue with the rest of this article.

Read Operation

Now I'll show you how to read the disk file you generated. You will read four records, the first, then the third, then back to the first, and finally the fifth logical record.

By typing in the name of the demonstration program, reenter the program. Do this, but with one additional entry, the name of the data file (TESTFILE/TST) following the name of the demonstration program.

For example, if the name of the demonstration program is DISKIO, type DISKIO TESTFILE/TST. You must include a single space between the program name DISKIO and the data file name. Every time you type in a command at TRSDOS READY, the HL register contains the address of the first non-blank character following the last command. The *Disk System Owner's Manual* describes this in detail under CMDTXT.

Once you type in the demonstration program name followed by the data file name, the program automatically saves the HL register in the location labeled PARAM. This brings you to the read section of the program at label GTSPC1, with an accompanying message. Then the program loads the HL register with the address contained in the location PARAM. The Read routine checks this address to see if it includes a carriage return, indicating the end of a command, and the absence of parameters to pass. If no parameters exist, the program asks you to enter the filespec just as when you initially ran the program.

"If you close the file successfully, the program sets the zero flag."

If you typed in the parameter TESTFILE/TST, the program uses this as the filespec to open the file. If you type an illegal parameter, an error occurs and you must enter a new filespec.

If everything goes right, the program opens the TESTFILE/TST file, ready to read. Before the program reads the file, it executes the FILPTR routine starting at label OPENOK. This routine requires that the DE register contain the DCB address. On return from FILPTR the B register contains the drive number on which the file resides, with the logical file number in the C register.

Continued on p. 100

Program Listing. Demonstration program.

```
;
;**********************************************************
;              FUNDAMENTAL ASSEMBLY LANGUAGE DISK I/O
;                       FOR THE TRS-80 MODEL III
;**********************************************************
;
         ORG     8000H
BKSPC    EQU     4445H              ;BACKSPACE 1 LOGICAL REC
CLOSE    EQU     4428H              ;CLOSE FILE ROUTINE
CMDDOS   EQU     429CH              ;EXEC TRSDOS COMMAND
CMDTXT   EQU     4225H              ;HAS LAST TRSDOS COMMAND
COMDOS   EQU     4299H              ;EXEC TRSDOS COMMAND
DIVIDE   EQU     4451H              ;DIVIDE ROUTINE
DSPDIR   EQU     4419H              ;DISPLAY DIRECTORY
ERRDSP   EQU     4409H              ;DISPLAY ERROR MESSAGE
FILPTR   EQU     428DH              ;FINDS FILE # AND DRIVE #
INIT     EQU     4420H              ;INITIALIZE DISK FILE
JP2DOS   EQU     402DH              ;ENTRY TO TRSDOS
KILL     EQU     442CH              ;KILLS FILE
OPEN     EQU     4424H              ;OPEN FILE ROUTINE
POSEOF   EQU     4448H              ;GO TO END OF FILE
POSN     EQU     4442H              ;POSITION TO LOGICAL REC.
PUTEXT   EQU     444BH              ;PUT EXTENSION IN DCB
RAMDIR   EQU     4290H              ;RAM DIRECTORY ROUTINE
READ     EQU     4436H              ;DISK READ ROUTINE
REWIND   EQU     443FH              ;REWIND FILE ROUTINE
SYNTAX   EQU     441CH              ;CHECK FILESPEC SYNTAX
VERF     EQU     443CH              ;WRITE AND VERIFY
WRITE    EQU     4439H              ;DISK WRITE ROUTINE
;
;**********************************************************
;              MISCELLANEOUS ROUTINES AND DEFINITIONS
;**********************************************************
;
KBLINE   EQU     40H                ;SCAN KEYBOARD FOR LINE
KBWAIT   EQU     49H                ;WAIT FOR KEYBOARD INPUT
LRL      EQU     128                ;LOGICAL RECORD LENGTH
LRL2     EQU     256                ;LOGICAL RECORD X2
VDCLS    EQU     1C9H               ;CLEAR SCREEN ROUTINE
VDLINE   EQU     21BH               ;DISPLAY LINE ROUTINE
;
;**********************************************************
;                          STORAGE LOCATIONS
;**********************************************************
;
ADRBUF   EQU     $                  ;ADDRESS OF 'BUFFER'
BUFFER   DEFS    256                ;DISK I/O BUFFER
AUREC    EQU     $                  ;ADDRESS OF 'UREC'
UREC     DEFS    640                ;USER BUFFER
ADRDCB   EQU     $                  ;ADDRESS OF 'DCB'
DCB      DEFS    64                 ;DATA CONTROL BLOCK
AKBBUF   EQU     $                  ;ADDRESS OF 'KBLNBF'
KBLNBF   DEFS    16                 ;KEYBOARD INPUT BUFFER
ADRDIR   EQU     $                  ;ADDRESS OF 'DIRBUF'
DIRBUF   DEFS    24                 ;DIRECTORY BUFFER
LENGTH   EQU     $-BUFFER
DRVNUM   DEFS    1                  ;ASCII CODED DRIVE NUMBER
LKBUF    DEFS    2
AENTRY   EQU     $                  ;ADDRESS OF RE-ENTRY BUF
ENTRY    DEFS    2                  ;RE-ENTRY ADDRESS BUFFER
PARAM    DEFS    2                  ;PARAMETER ADDRESS BUFFER
;
;**********************************************************
;                      WRITE DISK FILE EXERCISE
;**********************************************************
;
START    LD      (PARAM),HL         ;SAVE PARAMETER POINTER
         CALL    VDCLS              ;CLEAR SCREEN
         LD      HL,(ENTRY)         ;GET RE-ENTRY ADDRESS
         LD      A,H
         AND     0F0H
         CP      80H                ;IS THIS FIRST TIME THRU?
         JR      NZ,START1          ;YES
         JP      (HL)               ;NO. ENTER WHERE LEFT OFF
START1   CALL    CLRBUF             ;CLEAR OUT BUFFERS
         CALL    MOVDAT             ;FILL USER BUFFER W/DATA
         LD      HL,SIGNON          ;SIGNON MESSAGE
         CALL    VDLINE             ;DISPLAY MESSAGE
GETSPC   LD      HL,SPCMSG          ;FILESPEC MESSAGE
         CALL    VDLINE             ;DISPLAY MESSAGE
         LD      HL,KBLNBF          ;KEYBOARD INPUT BUFFER
         LD      B,15               ;MAX # OF CHARS. ALLOWED
         CALL    KBLINE             ;GET FILESPEC
         LD      HL,KBLNBF          ;FILESPEC ADDRESS
```

Listing continued


LOAD 80
LOAD 80
TO YOUR
RESCUE

See List of Advertisers on Page 307

Continued from p. 96

If FILPTR finds the appropriate information, it sets the zero flag. Otherwise, it resets the flag and the program skips down to the label READ01.

On successful completion of FILPTR, the routine RAMDIR uses the BC register, returned by FILPTR, to get the directory information of TESTFILE/TST and places it in memory.

RAMDIR requires that the BC register contain the drive number in the B register, the file number in the C register, and the address of a 24-byte buffer in the HL register. The 24-byte buffer in this case is DIRBUF. The *Disk System Owner's Manual* states that the buffer must be 22 bytes long for data, plus 1 byte for the plus symbol, which indicates the end of the directory. In fact, there are two plus marks at the end of the directory, thereby requiring 24 bytes in the buffer. If RAMDIR is successful, it sets the zero flag.

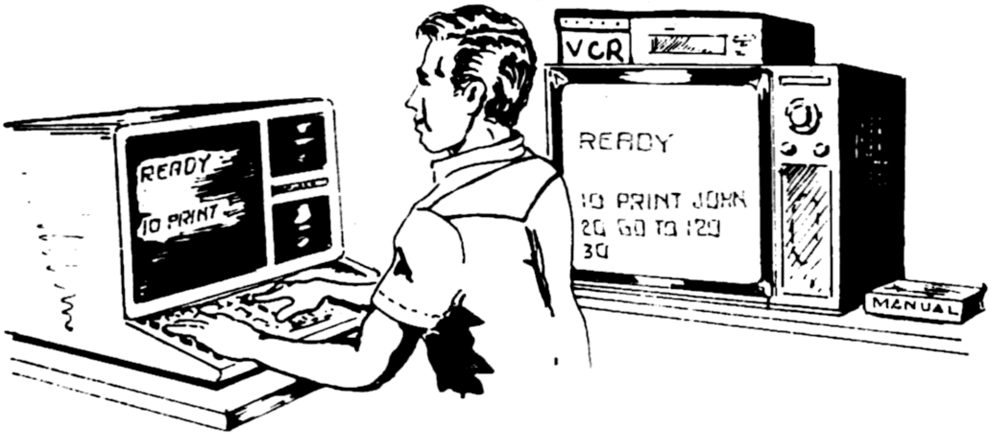
Following the execution of RAMDIR, the program converts the drive number and file number of TESTFILE/TST contained in the BC register to ASCII and places them in two separate messages. In addition, it saves the ASCII coded drive number in a location labeled DRVNUM for later use

Listing continued

```
                LD      DE,DCB          ;DATA CONTROL BLOCK
                CALL    SYNTAX          ;TEST FOR CORRECT SYNTAX
                JR      NZ,GETSPC       ;BAD SYNTAX
                CALL    LOOK            ;EXAMINE FILESPEC ?
        EXTEND  CALL    MSG001          ;DISPLAY ENTRY MESSAGES
                LD      HL,KBLNBF       ;KEYBOARD INPUT BUFFER
                LD      B,3             ;ALLOWS 3 CHAR. ENTRY
                CALL    KBLINE          ;GET EXTENSION
                LD      HL,KBLNBF       ;EXTENSION ADDRESS
                LD      DE,DCB          ;DATA CONTROL BLOCK
                CALL    PUTEXT          ;ADD EXTENSION TO FILSPC
                CALL    LOOK            ;EXAMINE FILESPEC ?
        OPNFIL  CALL    MSG002          ;DISPLAY ENTRY MESSAGES
                LD      HL,BUFFER       ;ADDR SECTOR I/O BUFFER
                LD.     DE,DCB          ;DATA CONTROL BLOCK
                LD      B,LRL           ;LOGICAL RECORD LENGTH
                CALL    OPEN            ;OPEN FILE
                JR      Z,INITOK        ;FILE OPENING SUCCESSFUL
                CALL    ERPROC          ;DISPLAY ERROR MESSAGE
                CALL    LOOK            ;EXAMINE DCB ?
        NEWFIL  CALL    MSG003          ;DISPLAY ENTRY MESSAGES
                LD      HL,BUFFER       ;ADDR SECTOR I/O BUFFER
                LD      DE,DCB          ;DATA CONTROL BLOCK
                LD      B,LRL           ;LOGICAL RECORD LENGTH
                CALL    INIT            ;INITIALIZE A NEW FILE
                JR      Z,INITOK        ;NEW FILE INITIALIZED
                CALL    ERPROC          ;DISPLAY ERROR MESSAGE
        INITOK  CALL    LOOK            ;EXAMINE DCB ?
        WRTFIL  CALL    MSG004          ;DISPLAY ENTRY MESSAGES
                LD      HL,UREC         ;USER BLOCK
                LD      DE,DCB          ;DATA CONTROL BLOCK
                CALL    VERF            ;WRITE DATA TO DISK
                JR      Z,WRTOK1        ;SUCCESSFUL WRITE
                CALL    ERPROC          ;DISPLAY ERROR MESSAGE
        WRTOK1  CALL    LOOK            ;EXAMINE BUFFERS ?
        WRT002  CALL    MSG005          ;DISPLAY ENTRY MESSAGES
                LD      HL,UREC+LRL     ;NEXT LRL OF DATA
                LD      DE,DCB          ;DATA CONTROL BLOCK
                CALL    WRITE           ;WRITE DATA TO DISK
                JR      Z,WRTOK2        ;SUCCESSFUL WRITE
                CALL    ERPROC          ;DISPLAY ERROR
        WRTOK2  CALL    LOOK            ;EXAMINE BUFFERS ?
        WRT003  CALL    MSG006          ;DISPLAY ENTRY MESSAGES
                LD      HL,UREC+LRL2    ;NEXT LRL OF DATA
                LD      DE,DCB          ;DATA CONTROL BLOCK
                CALL    WRITE           ;WRITE DATA TO DISK
                JR      Z,WRTOK3        ;SUCCESSFUL WRITE
                CALL    ERPROC          ;DISPLAY ERROR
        WRTOK3  CALL    LOOK            ;EXAMINE BUFFERS ?
        WRT004  CALL    MSG007          ;DISPLAY ENTRY MESSAGES
                LD      HL,UREC+LRL+LRL2
                LD      DE,DCB          ;DATA CONTROL BLOCK
                CALL    WRITE           ;WRITE DATA TO DISK
                JR      Z,WRTOK4        ;SUCCESSFUL WRITE
                CALL    ERPROC          ;DISPLAY ERROR
        WRTOK4  CALL    LOOK            ;EXAMINE BUFFERS ?
        WRT005  CALL    MSG008          ;DISPLAY ENTRY MESSAGES
                LD      HL,UREC+LRL2+LRL2
                LD      DE,DCB          ;DATA CONTROL BLOCK
                CALL    WRITE           ;WRITE DATA TO DISK
                JR      Z,WRTOK5        ;SUCCESSFUL WRITE
                CALL    ERPROC          ;DISPLAY ERROR
        WRTOK5  CALL    LOOK            ;EXAMINE BUFFERS ?
        CLOSFL  CALL    MSG009          ;DISPLAY ENTRY MESSAGES
                LD      DE,DCB          ;DATA CONTROL BLOCK
                CALL    CLOSE           ;CLOSE FILE
                JR      Z,CLOSOK        ;FILE CLOSED SUCCESSFULLY
                CALL    ERPROC          ;DISPLAY ERROR MESSAGE
        CLOSOK  LD      HL,GTSPC1       ;GET RE-ENTRY ADDRESS
                LD      (ENTRY),HL      ;SET NEW RE-ENTRY ADDRESS
                JP      JP2DOS          ;JUMP TO TRSDOS READY
        ;
        ;*******************************************************
        ;                  READ DISK FILE EXERCISE
        ;*******************************************************
        ;
        GTSPC1  CALL    CLRBUF          ;CLEAR OUT BUFFERS
                CALL    MSG010          ;DISPLAY ENTRY MESSAGES
                LD      HL,(PARAM)      ;GET PARAMETER ADDRESS
                LD      A,(HL)          ;FIRST CHARACTER OF PARAM
                CP      0DH             ;CARRIAGE RETURN ?
                JR      NZ,RDOPEN       ;OPEN FILE USING PARAM.
        GTSPC2  LD      HL,SPCMSG       ;GET FILESPEC MESSAGE
                CALL    VDLINE          ;DISPLAY MESSAGE
                LD      HL,KBLNBF       ;KEYBOARD INPUT BUFFER
                LD      B,15            ;MAX # CHARS FROM KEYBRD
                CALL    KBLINE          ;INPUT KEYBOARD DATA
                LD      HL,KBLNBF       ;FILESPEC
        RDOPEN  LD      DE,DCB          ;DATA CONTROL BLOCK
                CALL    SYNTAX          ;CHECK FOR CORRECT SYNTAX
                JR      NZ,GTSPC2       ;BAD SYNTAX
                LD      HL,BUFFER       ;SECTOR I/O BUFFER
                LD      DE,DCB          ;DATA CONTROL BLOCK
                LD      B,LRL           ;LOGICAL RECORD LENGTH
                CALL    OPEN            ;OPEN FILE
                JR      Z,OPENOK        ;OPENED SUCCESSFULLY
                CALL    ERPROC          ;DISPLAY ERROR MESSAGE
                JR      GTSPC2          ;GET NEW FILESPEC
        OPENOK  LD      DE,DCB          ;DATA CONTROL BLOCK
                CALL    FILPTR          ;GET DRV/FILE # IN BC REG
                JR      NZ,READ01       ;SKIP IF ERROR
                LD      HL,DIRBUF       ;DIRECTORY BUFFER
                CALL    RAMDIR          ;GET DIRECTORY INFO
                JR      NZ,READ01       ;SKIP IF ERROR
                LD      A,B             ;PUT DRIVE # IN A REG
                ADD     A,30H           ;MAKE ASCII
                LD      (DRVNUM),A      ;SAVE FOR ROUTINE DSPDIR
```

Listing continued

VIDEO INSTRUCTION TAPES!



THE OLD HARD WAY

THE NEW EASY WAY

STEP BY STEP INSTRUCTIONS

PICTURES ARE WORTH
THOUSANDS OF WORDS AND SAVE
HOURS OF FRUSTRATION

Use your VCR side by side with your computer to learn disk operating systems, how to program, and how to use programs. Your VCR along with your computer serve as your personal tutor. Pause your VCR to review and learn at your own pace.

VHS or BETA FORMAT

SPECIAL SALE WHILE SUPPLY LAST

Verbatim Datalife Diskettes
5 Year Warranty 5¼ In
Double Density Diskettes In
Plastic Storage Box

ONLY $25.00 BOX OF TEN

MASTERCARD — VISA and C.O.D. Orders accepted add $3.00 per order for shipping and handling. Telephone orders taken 9:00 a.m. - 6:00 p.m. Central Time.
Specify either VHS or BETA Tape Format

C.O.D. Orders add $3.00

DEALER INQUIRIES WELCOME

TO ORDER PHONE
OR WRITE

TAPES NOW AVAILABLE

CAT #	TOPIC	APPROX RUN TIME
DOS-1	TRSDOS 1.3 MOD 3	1 HR 30 MIN
DOS-6	TRSDOS 6.0 MOD 4	1 HR 45 MIN
DOS-3	NEWDOS 80 2.0	1 HR 45 MIN
DOS-2	DOSPLUS 3.5	1 HR 45 MIN
DOS-5	LDOS	1 HR 45 MIN
DOS-4	MULTIDOS	1 HR 45 MIN
EW-1	VISICALC	1 HR 30 MIN
WP-1	LAZY WRITER	1 HR 45 MIN
WP-2	SUPERSCRIPSIT	1 HR 45 MIN
WP-3	SCRIPSIT	1 HR 30 MIN
DB-1	PROFILE III PLUS	1 HR 45 MIN
DIO-3	TRSDOS 1.3 DISK I/O	1 HR 45 MIN
DIO-4	TRSDOS 6.0 DISK I/O	1 HR 45 MIN

The DIO-3 and DIO-4 Video Tapes include disk basic commands with examples. Also teaches both random and sequential disk read and write.

TAPE ABOVE ONLY $39.95 EACH

CAT #	TOPIC	APPROX RUN TIME
BP-1	LEARNING MODEL 3 BASIC	3 HR
BP-2	LEARNING MODEL I BASIC	3 HR
BP-3	LEARNING C-64 BASIC	2 HR
BP-4	LEARNING VIC-20 BASIC	2 HR
DIO-1	COMMODORE 64 DISK I/O	1 HR 45 MIN
DIO-2	VIC 20 DISK I/O	1 HR 45 MIN

TAPES ABOVE ONLY $49.95 EACH

LYNN COMPUTER SERVICE

6831 West 157th Street Tinley Park, Illinois 60477
(312) 429-1915

✓115

VISICALC IS A TRADEMARK OF PERSONEL SOFTWARE INC. LAZY WRITER IS A TRADEMARK OF ALPHA BIT COMMUNICATION SCRIPSIT, SUPERSCRIPSIT, PROFILE III PLUS AND TRSDOS ARE TRADEMARKS OF TANDY CORP. LDOS IS A TRADEMARK OF LOGICAL SYSTEMS INC. NEWDOS80 IS A TRADEMARK OF APPARAT INC. DOS PLUS IS A TRADEMARK OF MICRO-SYSTEMS SOFTWARE INC. MULTIDOS IS A TRADEMARK OF COSMOPOLITAN INC. VIC 20 AND COMMODORE 64 ARE TRADEMARKS OF COMMODORE BUSINESS MACHINES, INC.

✓ See List of Advertisers on Page 291

Listing continued

```
        LD      (DRVBUF),A      ;PUT IN MESSAGE
        LD      L,C             ;SET UP HL REGISTER TO
        LD      H,0             ;  MAKE FILE NUMBER AN
        LD      A,10            ;  ASCII STRING.
        CALL    DIVIDE
        ADD     A,30H           ;MAKE REMAINDER ASCII
        LD      (FILNUM+1),A    ;SAVE IN MESSAGE
        LD      A,L             ;GET REST OF FILE NUMBER
        ADD     A,30H           ;MAKE IT ASCII
        LD      (FILNUM),A      ;PUT IN MESSAGE
        LD      HL,DRVMSG       ;DRIVE NUMBER MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,FILMSG       ;FILE NUMBER MESSAGE
        CALL    VDLINE          ;DISPLAY REST OF MESSAGE
        CALL    LOOK            ;EXAMINE DIRECTORY ?
READ01  CALL    MSG011          ;DISPLAY ENTRY MESSAGES
        LD      HL,UREC         ;USER DATA BUFFER
        LD      DE,DCB          ;DATA CONTROL BLOCK
        CALL    READ            ;READ A LOGICAL RECORD
        JR      Z,READOK        ;RECORD READ OK
        CALL    ERPROC          ;DISPLAY ERROR MESSAGE
READOK  CALL    LOOK            ;EXAMINE BUFFERS ?
POSENT  CALL    MSG012          ;DISPLAY ENTRY MESSAGES
        LD      DE,DCB          ;DATA CONTROL BLOCK
        LD      BC,2            ;LRL TO GO TO
        CALL    POSN            ;GO TO 3RD LOGICAL RECORD
        JR      Z,POSNOK        ;POSITIONED OK
        CALL    ERPROC          ;DISPLAY ERROR MESSAGE
POSNOK  CALL    LOOK            ;EXAMINE DCB ?
READ02  CALL    MSG013          ;DISPLAY ENTRY MESSAGES
        LD      HL,UREC+LRL     ;OFFSET INTO BUFFER 1 LRL
        LD      DE,DCB          ;DATA CONTROL BLOCK
        CALL    READ            ;READ NEXT LOGICAL RECORD
        JR      Z,RDOK          ;NEXT RECORD READ OK
        CALL    ERPROC          ;DISPLAY ERROR MESSAGE
RDOK    CALL    LOOK            ;EXAMINE BUFFERS ?
REWENT  CALL    MSG014          ;DISPLAY ENTRY MESSAGES
        LD      DE,DCB          ;DATA CONTROL BLOCK
        CALL    REWIND          ;GO TO BEGINNING OF FILE
        JR      Z,REWOK         ;REWIND COMPLETE
        CALL    ERPROC          ;PROCESS ERROR MESSAGE
REWOK   CALL    LOOK            ;EXAMINE BUFFERS ?
READ03  CALL    MSG015          ;DISPLAY ENTRY MESSAGES
        LD      HL,UREC+LRL2    ;NEXT LOGICAL RECORD
        LD      DE,DCB          ;DATA CONTROL BLOCK
        CALL    READ            ;READ FIRST LOGICAL REC
        JR      Z,RDOK1         ;FIRST LOG REC READ OK
        CALL    ERPROC          ;PROCESS ERROR MESSAGE
RDOK1   CALL    LOOK            ;EXAMINE BUFFERS ?
EOFENT  CALL    MSG016          ;DISPLAY ENTRY MESSAGES
        LD      DE,DCB          ;DATA CONTROL BLOCK
        CALL    POSEOF          ;GO TO END OF FILE
        JR      Z,EOFOK         ;POSITIONED TO EOF OK
        CALL    ERPROC          ;PROCESS ERROR MESSAGE
EOFOK   CALL    LOOK            ;EXAMINE BUFFERS ?
BKENT   CALL    MSG017          ;DISPLAY ENTRY MESSAGES
        LD      DE,DCB          ;DATA CONTROL BLOCK
        CALL    BKSPC           ;BACK UP 1 LOGICAL RECORD
        JR      Z,BKSPOK        ;BACKSPACE 1 LRL OK
        CALL    ERPROC          ;PROCESS ERROR MESSAGE
BKSPOK  CALL    LOOK            ;EXAMINE BUFFERS ?
READ04  CALL    MSG018          ;DISPLAY ENTRY MESSAGES
        LD      HL,UREC+LRL2+LRL        ;USER BUFFER
        LD      DE,DCB          ;DATA CONTROL BLOCK
        CALL    READ            ;READ LOGICAL RECORD
        JR      Z,RDOK2         ;READ SUCCESSFUL
        CALL    ERPROC          ;PROCESS ERROR MESSAGE
RDOK2   CALL    LOOK            ;EXAMINE BUFFERS ?
CLSENT  CALL    MSG019          ;DISPLAY ENTRY MESSAGES
        LD      DE,DCB          ;DATA CONTROL BLOCK
        CALL    CLOSE           ;CLOSE FILE
        JR      Z,KILFIL        ;CLOSE OF FILE OK
        CALL    ERPROC          ;PROCESS ERROR MESSAGE
KILFIL  LD      HL,KILMSG       ;KILLING FILE MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      DE,DCB          ;DATA CONTROL BLOCK
        CALL    KILL            ;KILL FILE
        JR      Z,KILLOK        ;KILL SUCCESSFUL
        CALL    ERPROC          ;PROCESS ERROR
KILLOK  LD      A,(DRVNUM)      ;GET DRIVE NUMBER
        LD      (4271H),A       ;PUT IN SYSTEM AREA
        CALL    DSPDIR          ;DISPLAY DIRECTORY
        LD      HL,ENDMSG       ;END OF DEMO MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        CALL    KBWAIT          ;WAIT FOR KEYBOARD ENTRY
        JP      JP2DOS          ;GO TO TRSDOS READY
;
;**********************************************************
;                   ERROR DISPLAY PROCESSOR
;**********************************************************
;
ERPROC  OR      0C0H            ;DISPLAY MODE FOR ERRDSP
        CALL    ERRDSP          ;DISPLAY ERROR MESSAGE
        RET
;
;**********************************************************
;                   EXAMINE DATA PROCESSOR
;**********************************************************
;
LOOK    POP     HL              ;GET RETURN ADDRESS
        PUSH    HL              ;RESTORE IT
        LD      (ENTRY),HL      ;SAVE RETURN ADDRESSS
LOOK1   LD      HL,LKMSG        ;LOOK? MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,LKBUF        ;ANSWER BUFFER
        LD      B,1             ;ALLOW 1 CHAR TO RTN
```

Listing continued

by the system routine DSPDIR.

The program displays the messages and calls Look so you can see the directory in RAM. You can use the directory to find out how many records are in the file, the file's logical record length, and other information needed to read the file. See Table 4 for the directory's layout.

Since you already know the number of logical records and the logical record length, returning to the demonstration program after looking at the directory leads right into reading the file starting at label READ01.

When you open a file, the data in the DCB points to the first logical record in the file. A message indicates that you are reading the first logical record. With the HL register pointing to UREC and the DE register pointing to the DCB, the program makes a call to Read. If the read is successful, it sets the zero flag and puts the first logical record into UREC. This gives you an opportunity to look at the buffer UREC.

After looking at UREC, return to the demonstration program. You should reenter at the label POSENT where the routine POSN executes so you can read the third logical record. POSN requires that the BC register point to the logical record to which you want to read or write. Since the logical records start with record zero, record 2 is the third logical record, therefore BC equals 2.

"You can use the directory to find out how many records are in the file."

The DE register must point to the DCB. If the positioning is successful, the subroutine sets the zero flag and a Look call lets you view the DCB. You should note here that the next physical record number at bytes 10 and 11 in the DCB is now 1, and byte 5 (the offset to delimiter at end of current physical record) is zero. This confirms your position at the end of the second logical record and points to the third logical record.

Now return to the demonstration program reentering at the label READ-02. Here the program reads the third logical record. The program loads the HL register with UREC plus LRL, allowing the next read from the disk to place the data in the next 128 bytes of the user buffer UREC. The program next loads the DE register with the DCB address and executes a Read command. On successful completion, a call to Look lets you examine UREC and the DCB.

Now return to the demonstration program at label REWENT where it executes a Rewind routine. This routine requires that the program load only the DE register with the address of the DCB. Once the program executes Rewind, the DCB points to the beginning of the file, record zero. Again, a successful execution sets the zero flag and you can look at the DCB via Look.

Returning to the demonstration program should place you at the label READ03. Here the program reads the first logical record. The HL register points one more logical record length into UREC, and the DE register contains the DCB address. The call to Read reads the first logical record into UREC. Following a successful read, you can look at UREC to verify that it now contains the first, third, and first logical records of TESTFILE/TST.

You can do one of two things in order to read the last logical record. You can use the POSN routine to go to logical record 5, or you can go to the end of the file and backspace one logical record. Here I do the latter.

Return to the demonstration program at label EOFENT. The DE register contains the address of the DCB and the program calls POSEOF. If successful, it sets the zero flag and the DCB points to the beginning of the next logical record beyond the last logical record.

For a write operation, you can extend the file here. But since you're reading and you want to look at the last logical record, you must backspace one record. If you looked at the DCB after the program executes POSEOF, return to the demonstration program at label BKENT.

With the DE register pointing to the DCB address, a call to BKSPC repositions the pointer in the DCB to the last logical record. To allow examination of the DCB, the program calls the Look routine. Considering bytes 10 and 11, the next physical record is 2 and the offset to delimiter at end of current record at byte 5 is zero. This verifies that the DCB is pointing to the fifth logical record.

Continuing with the demonstration program at label READ04, it loads the HL register with the address of the next logical record position into UREC and places the address of the DCB in the DE

Listing continued

```
        CALL    KBLINE          ;GET CHARACTER
        JR      C,LOOK1         ;BREAK KEY PRESSED
        LD      A,(HL)          ;GET ANSWER
        CP      'Y'             ;WAS IT YES ?
        RET     NZ              ;NO
        LD      HL,EXECUT       ;ADDRESS OF COMMAND
        JP      COMDOS          ;ENTER DEBUG
;
;*********************************************************
;               ENTRY POINT MESSAGE DISPLAY ROUTINES
;*********************************************************
;
MSG001  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT001       ;ENTRY POINT NAME #1
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,EXTMSG       ;PROMPT FOR EXTENSION
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG002  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT002       ;ENTRY POINT NAME #2
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,OPNMSG       ;OPEN FILE MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG003  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT003       ;ENTRY POINT NAME #3
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,INITMS       ;INITIALIZING FILE MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG004  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DIPLAY MESSAGE
        LD      HL,ENT004       ;ENTRY POINT NAME #4
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,LRL1ST       ;'FIRST' LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,WRT001       ;WRITING LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG005  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT005       ;ENTRY POINT NAME #5
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,LRL2ND       ;'SECOND' LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,WRT001       ;WRITING LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG006  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT006       ;ENTRY POINT NAME #6
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,LRL3RD       ;'THIRD' LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,WRT001       ;WRITING LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG007  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT007       ;ENTRY POINT NAME #7
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,LRL4TH       ;'FOURTH' LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,WRT001       ;WRITING LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG008  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT008       ;ENTRY POINT NAME #8
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,LRL5TH       ;'FIFTH' LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,WRT001       ;WRITING LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG009  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT009       ;ENTRY POINT NAME #9
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,CLOSMS       ;CLOSING FILE MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG010  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT010       ;ENTRY POINT NAME #10
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,RDMSG        ;READING FILE MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG011  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT011       ;ENTRY POINT NAME #11
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,LRL1ST       ;'FIRST' LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,RDLRL        ;READING LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG012  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT012       ;ENTRY POINT NAME #12
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,POSMSG       ;POSITIONING MESSAGE
```

Listing continued

Listing continued

```
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG013  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT013       ;ENTRY POINT NAME #13
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,LRL3RD       ;'THIRD' LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,RDLRL        ;READING LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG014  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT014       ;ENTRY POINT NAME #14
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,REWMSG       ;REWINDING FILE MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG015  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT015       ;ENTRY POINT NAME #15
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,LRL1ST       ;'FIRST' LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,RDLRL        ;READING LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG016  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT016       ;ENTRY POINT NAME #16
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,EOFMSG       ;POSITIONING TO EOF MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG017  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT017       ;ENTRY POINT NAME #17
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,BKSPMS       ;BACKSPACING MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG018  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT018       ;ENTRY POINT NAME #18
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,LRL5TH       ;'FIFTH' LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,RDLRL        ;READING LOGICAL REC MSG
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
MSG019  LD      HL,ENTMSG       ;ENTRY POINT MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        LD      HL,ENT019       ;ENTRY POINT NAME #19
        CALL    VDLINE          ;DISPLAY IT
        LD      HL,CLOSMS       ;CLOSING FILE MESSAGE
        CALL    VDLINE          ;DISPLAY MESSAGE
        RET
;
;**********************************************************
;                       CLEAR BUFFER AREAS
;**********************************************************
;
CLRBUF  LD      BC,LENGTH       ;LENGTH OF STORAGE AREA
        LD      HL,BUFFER       ;START OF STORAGE AREA
CLRLP   XOR     A               ;CLEAR A REGISTER
        LD      (HL),A          ;CLEAR STORAGE AREA
        DEC     BC              ;DECREMENT BYTE COUNTER
        LD      A,B             ;IS CLEARING FUNC. DONE ?
        OR      C
        INC     HL              ;GO TO NEXT LOCATION
        JR      NZ,CLRLP        ;NO, FINISH CLEARING BUFS
        RET
;
;**********************************************************
;                   PROVIDE DUMMY DATA FOR WRITE
;**********************************************************
;
MOVDAT  LD      DE,UREC         ;ADDRESS OF USER BUFFER
        LD      A,8             ;LOOP COUNTER
MOVLP1  LD      HL,LRL001       ;DATA TO BE MOVED
        LD      BC,16           ;NUMBER OF BYTES TO MOVE
        LDIR                    ;MOVE THEM
        DEC     A               ;REDUCE LOOP COUNTER
        JR      NZ,MOVLP1       ;FINISH DATA MOVE
        LD      A,8             ;LOOP COUNTER
MOVLP2  LD      HL,LRL002       ;DATA TO BE MOVED
        LD      BC,16           ;NUMBER OF BYTES TO MOVE
        LDIR                    ;MOVE THEM
        DEC     A               ;REDUCE LOOP COUNTER
        JR      NZ,MOVLP2       ;FINISH DATA MOVE
        LD      A,8             ;LOOP COUNTER
MOVLP3  LD      HL,LRL003       ;DATA TO BE MOVED
        LD      BC,16           ;NUMBER OF BYTES TO MOVE
        LDIR                    ;MOVE THEM
        DEC     A               ;REDUCE LOOP COUNTER
        JR      NZ,MOVLP3       ;FINISH DATA MOVE
        LD      A,8             ;LOOP COUNTER
MOVLP4  LD      HL,LRL004       ;DATA TO BE MOVED
        LD      BC,16           ;NUMBER OF BYTES TO MOVE
        LDIR                    ;MOVE THEM
        DEC     A               ;REDUCE LOOP COUNTER
        JR      NZ,MOVLP4       ;FINISH DATA MOVE
        LD      A,8             ;LOOP COUNTER
MOVLP5  LD      HL,LRL005       ;DATA TO BE MOVED
        LD      BC,16           ;NUMBER OF BYTES TO MOVE
        LDIR                    ;MOVE THEM
```

Listing continued

register. The call to Read puts the fifth logical record into the user buffer UREC. A call to Look lets you view the new data in UREC.

With the reading of all the records complete, you can now reenter the program at label CLSENT, where a number of things happen in succession. You first want to close the file. Do so in the same manner in which you built the file, with the DE register pointing to the DCB and by executing a call to Close.

Since you no longer have any use for this file, you can also kill it. The only parameter you need to kill the file is to point the DE register to the DCB and make a call to the Kill routine. A successful kill operation returns the zero flag set.

To verify this, look at the disk directory to see that the directory no longer lists the file name. The subroutine KILL now sets the first 16 bytes of the DCB to zero.

The DSPDIR routine displays the disk directory. It needs only the drive number of the disk from which you want a directory, coded as an ASCII number in location 4271 hex. If you remember, you put the drive number in the location DRVNUM. Picking it up here and moving it to 4271 hex, followed by a call to DSPDIR, verifies that you killed TESTFILE/TST. The directory listing is abbreviated, showing only file names.

Now return to the demonstration program. Here, two messages indicate that you've closed and killed the file. Immediately following this, the program clears the screen and displays the directory.

Enter any character following the directory display to terminate the demonstration program with a jump to the TRSDOS entry point JP2DOS.

Two routines not covered in this program are VERF and CMDDOS. VERF is the same as Write except that after each write to disk, VERF reads the data to verify a proper write.

CMDDOS is similar to COMDOS except CMDDOS returns to the user's program after completing the routine. Of course, if you execute one of your programs from another program using this command, you must execute a final return to ensure that you make it possible to return. All the routines just covered are limited to accessing user generated files only. No system files are accessible. ■

Write to David G. Haan at 4361 S. Estes St., Littleton, CO 80123.

Listing continued

```
         DEC     A               ;REDUCE LOOP COUNTER
         JR      NZ,MOVLP5       ;FINISH DATA MOVE
         RET
;
;*************************************************************
;                        MESSAGES
;*************************************************************
;
SIGNON   DEFM    'ASSEMBLY LANGUAGE DISK I/O '
         DEFM    'FOR THE TRS-80 MODEL III.'
         DEFB    0DH
SPCMSG   DEFM    'ENTER FILENAME.'
         DEFB    0DH
LKMSG    DEFM    'DO YOU WISH TO LOOK AT ANY DATA?'
         DEFB    0DH
LRL001   DEFM    'LRL NUMBER ONE. '
LRL002   DEFM    'LRL NUMBER TWO. '
LRL003   DEFM    'LRL NUMBR THREE '
LRL004   DEFM    'LRL NUMBER FOUR '
LRL005   DEFM    'LRL NUMBER FIVE '
EXTMSG   DEFM    'ENTER 3 CHARACTER EXTENSION (E.G. TST).'
         DEFB    0DH
OPNMSG   DEFM    'ATTEMPTING TO OPEN FILE.'
         DEFB    0DH
INITMS   DEFM    'INITIALIZING NEW FILE.'
         DEFB    0DH
LRL1ST   DEFM    'FIRST '
         DEFB    03H
LRL2ND   DEFM    'SECOND '
         DEFB    03H
LRL3RD   DEFM    'THIRD '
         DEFB    03H
LRL4TH   DEFM    'FOURTH '
         DEFB    03H
LRL5TH   DEFM    'FIFTH '
         DEFB    03H
WRT001   DEFM    'LOGICAL RECORD BEING WRITTEN.'
         DEFB    0DH
CLOSMS   DEFM    'CLOSING FILE.'
         DEFB    0DH
RDMSG    DEFM    'THIS SECTION WILL INVOLVE '
         DEFM    'READING RANDOM LOGICAL RECORDS.'
         DEFB    0DH
RDLRL    DEFM    'LOGICAL RECORD BEING READ.'
         DEFB    0DH
POSMSG   DEFM    'POSITIONING TO THIRD LOGICAL RECORD.'
         DEFB    0DH
BKSPMS   DEFM    'BACKSPACING ONE LOGICAL RECORD.'
         DEFB    0DH
DRVMSG   DEFM    'FILE EXISTS ON DRIVE '
DRVBUF   DEFB    0
         DEFB    0DH
FILMSG   DEFM    'FILE NUMBER IS '
FILNUM   DEFW    0
         DEFB    0DH
REWMSG   DEFM    'REWINDING FILE.'
         DEFB    0DH
EOFMSG   DEFM    'POSITIONING TO END OF FILE.'
         DEFB    0DH
KILMSG   DEFM    'KILLING FILE.'
         DEFB    0DH
ENDMSG   DEFM    'END OF DEMONSTRATION PROGRAM...'
         DEFM    'PRESS ANY KEY TO CONTINUE.'
         DEFB    0DH
ENTMSG   DEFM    'REENTRY POINT IS AT PROGRAM LABEL '
         DEFW    03H
ENT001   DEFM    'EXTEND.'
         DEFW    0D0AH
ENT002   DEFM    'OPNFIL.'
         DEFW    0D0AH
ENT003   DEFM    'NEWFIL.'
         DEFW    0D0AH
ENT004   DEFM    'WRTFIL.'
         DEFW    0D0AH
ENT005   DEFM    'WRT002.'
         DEFW    0D0AH
ENT006   DEFM    'WRT003.'
         DEFW    0D0AH
ENT007   DEFM    'WRT004.'
         DEFW    0D0AH
ENT008   DEFM    'WRT005.'
         DEFW    0D0AH
ENT009   DEFM    'CLOSFL.'
         DEFW    0D0AH
ENT010   DEFM    'GTSPC1.'
         DEFW    0D0AH
ENT011   DEFM    'READ01.'
         DEFW    0D0AH
ENT012   DEFM    'POSENT.'
         DEFW    0D0AH
ENT013   DEFM    'READ02.'
         DEFW    0D0AH
ENT014   DEFM    'REWENT.'
         DEFW    0D0AH
ENT015   DEFM    'READ03.'
         DEFW    0D0AH
ENT016   DEFM    'EOFENT.'
         DEFW    0D0AH
ENT017   DEFM    'BKENT.'
         DEFW    0D0AH
ENT018   DEFM    'READ04.'
         DEFW    0D0AH
ENT019   DEFM    'CLSENT.'
         DEFW    0D0AH
EXECUT   DEFM    'DEBUG'
         DEFB    0DH
         END     START
```


See List of Advertisers on Page 291

WHAT DO YOU GET WITH THE PRODUCER?

You will be impressed with the professionalism of the PRODUCER package:

DISKETTE(s) containing PRODUCER Program Development System.

REFERENCE MANUAL of over 200 pages of extensive, easy to read, well organized material. Attractive hardback 3-ring binder. Color keyed index tabs separate the chapters. Comprehensive alphabetical Index refers to specific chapter subsections.

QUICK REFERENCE CARD

REGISTRATION CARD

TUTORIAL SESSION including audio cassettes and detailed follow-along outline, written and produced by fellow PRODUCER user.

FREE HOME INVENTORY MANAGEMENT PROGRAM ($59.95 value as a sample) allowing you to use a finished program immediately.

ONE YEAR SUBSCRIPTION to the PRODUCER newsletter

TOLL FREE NUMBER for technical assistance, available only to registered PRODUCER owners.

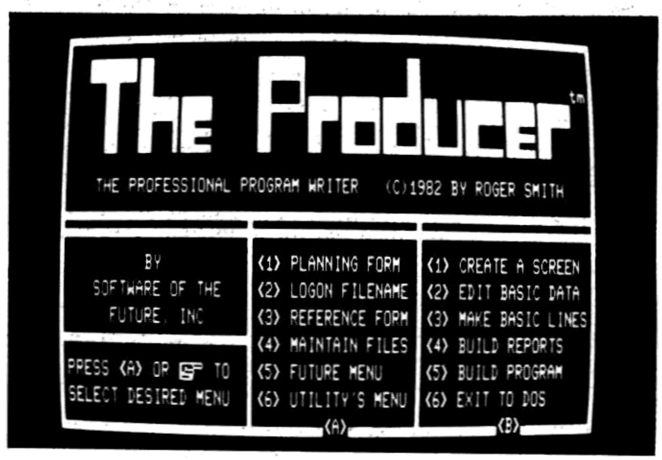
Pictured are the components of the Model III version of The PRODUCER. Other versions may vary slightly.

The PRODUCER

59

HOW THE PRODUCER WORKS

We think you will be impressed with the ease of operation and the amazing versatility of features you get with the PRODUCER. Here is a step by step overview of the program writing process. The screen shown is an unretouched photo of the Master Menu from which each of these steps is selected.



☐ Planning Your Program

The PRODUCER provides a helpful planning form you can print on your own printer. It helps you organize your thoughts to create a tailor made program to meet your needs.

☐ Creating The Screen

Visible on your monitor will be the screen where information will be entered, edited and displayed. There are six simple steps to follow in creating your screen.

1. Draw Your Screen

Using the arrow keys construct the screen in any configuration you desire. With single keystrokes, enter large graphic letters and borders. Edit at will until you are satisfied.

2. Define Message Areas

Select an area of your screen where The PRODUCER messages to you will appear.

3. Define Input Fields

The PRODUCER will ask you questions about the areas where you will enter the data. You specify the length of each area or field, as well as acceptable characters in each field.

4. Define Display Fields

Locate the display fields anywhere you want on your screen. These show the results of the calculations you want made on your data.

5. Define Custom Prompts

You select an area where help messages to yourself can be displayed.

6. Save Your Results

Assign a working name for your program and save it to disk.

☐ Editing Basic Data

1. Edit any part of The PRODUCER program you have created -- screen field names, lengths, prompt areas, etc.
2. Type in any help message you want as a custom prompt to help you operate the program.
3. Easily create calculations for your program using actual field names. You can use the contents of any numeric field and all math operations including logical operators.

☐ Making Basic Code

Press a key, sit back and watch The PRODUCER do all the work of creating BASIC code for your program. You can see the program lines appear on your screen. Complete error checking is done for you.

☐ Building Reports

Virtually any report is available to you thru our NEW free form report generator. It works with any size paper. You are allowed up to 100 calculations within the report. You can specify exact position of any text information to any position on your paper (even preprinted forms, checks, etc.). An amazingly versatile tool.

☐ Building The Program

Put the finishing touches on your program by selecting cursor type, size, flashing speed, auto messages, custom logos, etc. After your selections have been made, press a key and your entire finished program is created in less than 5 minutes. That's all there is to this remarkably simple program generation process.

Continued

The PRODUCER

TECHNICAL INFORMATION

The PRODUCER provides many advanced features which allow you to do "magic" with the programs you create.

The SCREEN GENERATOR

- Use the full screen (all lines and column positions)
- Create a professional well organized screen with graphics
- Save up to 9 separate screens in memory at one time and get instant access to each
- Move the cursor to any location on the screen
- Replicate bars/lines/graphics to define certain screen areas
- Access an instantly available Help Menu of all Screen Editor commands
- Insert and delete any character with a single keystroke
- Clear or erase selected areas of any screen
- Insert and delete whole lines on the screen
- Center any text on the screen
- Move any rectangular block of text anywhere on the screen (block move)
- Create titles with a single keystroke large graphic letter alphabet
- Move portions of screens between different screens (cut and paste)
- Save any number of screens to disk at any time
- Recall any screen from disk any time
- Create BASIC lines to re-create any screen

FILE and RECORD HANDLING

- Rapidly access records with BTREE File structure
- Search for a record with only the first few letters of the name or key (partial key) (Example: locate PRODUCER by typing PR)
- Recall and edit duplicate and multiple keys (Example: Several last names may be the same on a file and you can find and edit them individually
- Fully edit any part of a previously entered record
- Recover unused space automatically upon deletion of a record
- Enter data very fast with the special batch mode
- Recall immediately any record after it's been entered, eliminating time consuming sorting and indexing
- Rapidly access any record anytime (2-4 seconds average)
- Globally search and replace data in certain fields in selected record range
- Automatically rebuild any file to meet new specifications. No need to re-enter data when a file needs to be restructured.
- Balance any BTREE file automatically to reorganize and speed up file access time
- Recover from power failure and easily rebuild files that have been damaged. Avoid laborious re-entry of long data files

SCREEN ORIENTED INPUT and EDITING of DATA

- Insert and delete characters at any position in any field. No "back to start" retyping of data
- Move forward or back to previously entered fields to edit using the arrow keys. Totally non-destructive cursor. Does not require re-entering of each data field
- Move within any field using the arrow keys
- Move instantly to any field with Control G command
- Exit from input/edit mode at any point allowing immediate escape from data entry mode. Allows partial information to be entered for each record without the annoying, time consuming need to press ENTER for each blank field not used at the time of entry
- Duplicate field information from a previous record with one keystroke. No need to re-enter duplicate information, addresses, etc. on consecutive records
- View a custom prompt, your own custom reminder or help message for each field with 1 keystroke
- Verify each character typed automatically
- Enter data as fast as you want, even if you are a speed typist
- View visible display of automatic field length restrictions
- View prompts for each field showing number of characters allowed

PRINTED REPORTS

- Create up to 9 separate reports at a time in a finished program
- Generate any number of reports you want (no limit)
- Select reports by name from a report menu in the program
- Select from six different automatic report formats including custom mailing labels
- Instantly print reports by key with no time consuming sort necessary
- Sort and print any other (non key) field with the fast machine language sort
- Sort only records that meet your search criteria
- Sort on more than one field if desired
- Use any restrictions or search criteria to determine which records will be included in a report
- Use any number of multiple search criteriea (including logical) (Example: You can search for all the males who are single, and drive a car that are over 24 years old but less than 35 years old
- Send any special command to your printer before or after any report
- Specify any line length needed and any page length desired
- Select single line or multiple lines per record, even one page per record
- Total any fields during the report (running totals)

FREEFORM REPORT GENERATOR

Optional at only $49.95

- Specify column and row of every heading and field
- Allow up to 100 of interfield calculations, even string calculations
- Include any text anywhere on the screen
- Keep sub-totals on any field and print at any time in any format
- Format any numeric fields anyway you wish
- Print reports on pre-printed forms, checks, etc.
- Create form letters with merged field data, with no word processing necessary
- Put any field anywhere on the page. No limitations

ADVANCED CALCULATIONS

- Globally recalculate any field in any or all records. (Example: If file is a list of gold assets and the spot price changes, each separate asset may be recalculated with a new value for the spot price)
- Use all math operations including exponentiation and trigonometry
- Use logical calculations such as And, Or, Not, etc.
- Use any level of parenthesis in calculation formulas
- Save results in any field and display results in any field
- Store temporary results in several extra memory slots
- Pass calculation results between records
- Determine the exact order of calculations
- Display or save results at your option in the finished record

OTHER ADVANCED FEATURES

- Edit any part of any program without starting over or redefining the entire program
- Create screen and input modules only (for professional programmers)
- Create Calculate-only programs with the easy desk-top super calculator program
- Design custom logos for your program
- Control cursor type, size, flash speed, etc.
- Design custom prompts or help info for any field

YOU ALSO GET

- FREE 1 year SUBSCRIPTION to PRODUCER Newsletter
- TOLL FREE assistance number for all registered users
- REFERENCE MANUAL of over 200 pages
- FREE audio TUTORIAL

The PRODUCER

WHAT ARE PRODUCER USERS SAYING?

We continue to receive testimonials from satisfied users almost every day. Here's a sampling of the feedback we are receiving:

VALUE

VERY impressive! No matter how much I use the PRODUCER, there is no doubt I got my money's worth. It is clear the program, packaging and tutorial are developed with lots of thought....Very user friendly! Congratulations!
R. N. Forbes, Los Altos Hills, California

The PRODUCER package I received was excellent. The finest software package I have ever purchased. Far beyond my expectations.
S. R. Foster, Pensacola, Florida

I think the PRODUCER will save me so much time that it will give me the time to do the more important tasks that my business calls for and the money I'll save from not having to buy canned programs that are overpriced. Now with the PRODUCER I can write a program overnight to do almost anything I want it to do and with written reports to boot. Talk about saving time and money. I feel the PRODUCER will pay for itself with my first three programs.
S. Tornatore, Canastota, New York

The PRODUCER is a very impressive software package. It is well worth the money. While other micro owners are printing mailing labels, I am now selling them programs to use. I now have more time to spend enjoying my computer.
V. E. Ryberg, Bloomington, Illinois

I'm in love with the PRODUCER. It's one of my favorite programs.
R. Selsback, Burlingame, California

It was very complete and professionally done. The packaging and program seem to have been thought out before assembly and sale. The 'value' of the deal, everything included was the best I've seen to date.
G. Slusher, Martin, Kentucky

Very professional packaging. It gave the feeling of getting your money's worth before even running the program...Very easy to use and leaves very few questions unanswered...As you can see, I like the PRODUCER and was impressed with how trouble free it is.
A. C. Vincent, Napa, California

Excellent. Above and beyond other software.
R. Hapgood, Henrietta, Texas

VERSATILITY

The PRODUCER is the best all purpose program generator I have used. (We have tried almost all of them.) The generated code is bug free, well commented and efficient.
R. A. Copella, Northbrook, Ilinois

I bought the PRODUCER to save time. I feel capable of being able to write almost all programs I need. The PRODUCER generated programs will save a lot of time writing basic code and debugging. Using the PRODUCER I can write a good database type program using math calculation in about three hours. I don't have to tell you how long it would take writing the same program from scratch.
S. Tornatore, Canastota, New York

A special thanks to Roger and all of you. You've made my computing life easier and better. My 10 year old can't wait to get his hands on the PRODUCER.
J. D. Konkler, Columbus, Ohio

DOCUMENTATION

The Reference Manual is a work of art. Not only is it attractive and easy to use, it is so well organized, documented and logically written that the manual is a rarity in the software market place.
S. R. Foster, Pensacola, Florida

One of the best I've seen. We write about 20 volumes of material per year. Take it from a 'pro', it's good!
J. Crespi, Sherman Oaks, California

The PRODUCER Reference Manual is professionally written to provide ready acess to easily understood answers to questions which arise during use of the PRODUCER.
R. A. Copella, Northbrook, Illinois

The Reference Manual is supreme and superior to anything I have worked with.

R. A. Neuman, Okemos, Michigan

Very well laid out and organized. One of the best I've seen.
J. D. Konkler, Columbus, Ohio

QUALITY

Thank you for an excellent program. I agree that The PRODUCER will change the entire concept of program creation in the future. But for now, you stand as the best data-base-management-system I can buy.
E. Sung, Vancouver, B.C.

Your system really is Software of the Future. Your staff has insight others of us only dream of. Congratulations on a product of extraordinary design.
S. R. Foster, Pensacola, Florida

This is an excellent program. At this point I am totally pleased. This is by far my number one software and I will use it anywhere and everywhere I possibly can both personal and business. Once again congratulations to all of the people involved.
R. A. Neuman, Okemos, Michigan

Comparison shopping indicates the PRODUCER's superiority to all others. And I already own most of the others.
R. A. Copella, Northbrook, Ilinois

Glad to see you take an interest in what some of us hackers are up against. I think the PRODUCER will make the software hackers upgrade their products to this high level quality of the PRODUCER. I'm sure you realize that there is a lot of garbage on the market.
D. J. Smith, Lombard, Ilinois

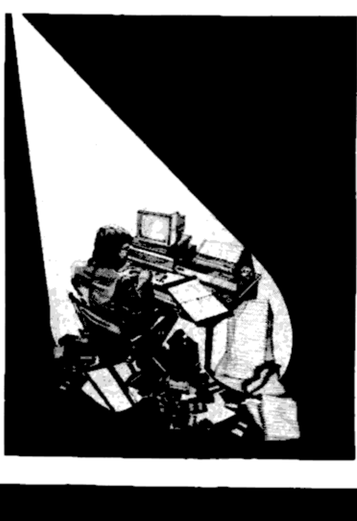
I was impressed by the professional appearance of your program. Other software I have received were on copy paper and stapled into a booklet with very vague instructions.
W. J. Mahaffey, Absecon, N. J.

USE

The program is almost idiot proof.
J. Crespi, Sherman Oaks, California

It is a very friendly friend and we will be working together for some time to come.
R. A. Neuman, Okemos, Michigan

Very easy to use and leaves very few questions unanswered.
A. C. Vincent, Napa, California



The PRODUCER

MODEL I version $149.95
MODEL III version $149.95

Available FALL/83 for

MODEL II $299.95
MODELIV $199.95
MODEL 12 .... $299.95
MODEL 16 .... $299.95
MODEL 16 XENIX $499.95
IBM - PC............. $299.95

PRODUCER SOFTWARE
Box 1245
Arlington, Texas
76004-1245

Texas 817-274-6998
800-433-5355

Finding the Search Solution

by Joseph E. Trojak

These two Assembly-language programs use signature screening techniques that help you easily maintain and search large text files.

If you have large text files that require efficient indexing and searching methods, Mindex and Search can help. Potential applications for these programs include indexing personal literature collections, client or patient records, mailing lists, card catalogs, and so on.

Most information retrieval programs for microcomputers use sequential searching techniques that are exceedingly slow, or use indexing systems that require laborious coding of key words. Mindex and Search use a method that quickly searches large files of text and avoids the problem of coding key words: signature screening.

The first part of this article reviews file searching techniques and explains the signature screening method. The second part describes how to apply the signature screening method to 8-bit microcomputers, and lists the Assembly-language source code for Mindex and Search.

Both programs run on the Models II, 12, 16, and III with 48K RAM and one or more disk drives.

The Key Box

Models II and III
64K RAM
Assembly Language
Radio Shack Assembly Language Development System (ALDS) or EDTASM

First, a few basic definitions. A key is that part of a record that identifies the record during a search. Primary keys uniquely specify a record. For example, when each record in a file contains the name, address, age, and social security number of a client, the social security number is a primary key for each record. It lets you call up a single, unique record.

Neither the name, address, nor age is sufficient to uniquely specify a record, so each is termed a secondary key or attribute. To avoid confusion I'll use the term search key to refer to the item you're searching for, and the term record key to denote the part of the record you're searching.

File Searching Techniques

Sequential Search

This is the simplest search technique. You start at the beginning of a file and compare every record key in the file with the search key until you find a match or the end of the file.

An example of this searching technique is that employed by a slow-witted detective who has a telephone number and needs to find the name and address of the person who leases the telephone line. The detective gets a telephone book, starts at page one, and compares every telephone number with the one he is searching for until he finds the exact number.

If the detective is lucky, the name begins with A; if he is very unlucky, it begins with Z. Most detectives would just dial the number and ask questions.

Binary Attribute Search

In this search technique, each record has a number of attributes (secondary keys), each of which has only two possible values: yes or no (true or false). Given a set of attributes, you can find all the records with these attributes by

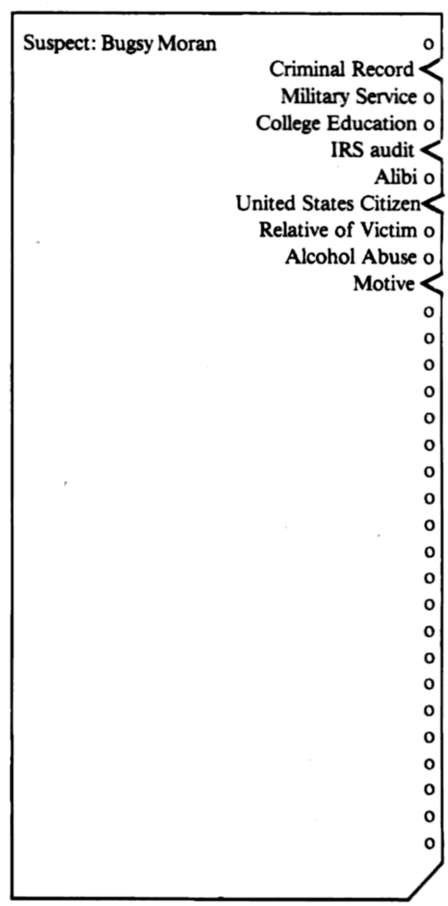


Figure 1. Edge-notched card for suspect Bugsy Moran.

comparing corresponding true and false values.

Our detective has a list of 200 suspects in a case on which he is working, and has the following information on each suspect: criminal record, military service, college education, IRS audit, alibi at time of crime, United States citizen, relative of victim, history of alcohol abuse, and motive for the crime. He wants to find all suspects who have a criminal record, no alibi, and a motive.

He goes to his local stationery store, purchases edge-notched cards, and codes them as shown in Fig. 1. Each attribute has a hole at the edge of the card. If an attribute is true, a notch is cut out of the card next to the attribute; otherwise the hole is left intact.

In Fig. 1, you can see that Bugsy Moran has a criminal record, no alibi, and a motive. Our detective makes his search in the following way: He arranges all 200 cards in a deck and puts a needle through the holes for the alibi attribute, then lifts the deck by raising the needle. All records of suspects with alibis fall out of the deck and he can exclude them.

The detective takes the cards that didn't fall out and puts needles into the holes for the criminal history and motive attributes. When he lifts the needles, all records of suspects with a criminal past, no alibi, and a motive for the crime fall out. This amounts to 10 cards; he has narrowed his search to 10 out of 200 suspects.

You can consider the holes and notches in the edge-notched card as a string of bits. For any attribute, the bit is either zero (the hole is intact) or 1 (the card has a notch). The attribute bit string of each record is called the record's signature and is an index of the record's attributes.

Binary Attribute Search of Superimposed Codes

This technique is similar to the simple binary attribute search except that the signature has more attributes than bits. The principle is to map a set of m binary attributes from a record into a set of n codes, then superimpose the codes for each attribute onto an n-bit signature.

Our detective is so enamored with his success in pruning his suspect list, he decides to use edge-notched cards for another purpose. In his state, automobile license numbers comprise three two-

Automobile Model	Notch Code
Buick	99 21
Cadillac	54 83
Chevrolet	41 78
Chrysler	07 31
DeSoto	67 81
Dodge	46 15
Edsel	66 10
Ford	73 09
Hudson	32 77
Lincoln	28 41
Mercury	12 88
Nash	34 89
Oldsmobile	59 42
Packard	44 75
Pontiac	74 28
Plymouth	57 71
Rambler	93 14
Rolls Royce	61 85
Studebaker	01 94
Volkswagen	57 39

Automobile Color	Notch Code
Black	40 74
Blue	02 77
Brown	04 14
Chartreuse	15 80
Gray	69 52
Green	23 94
Maroon	29 44
Red	39 15
White	70 96
Yellow	59 20

Table 1. Superimposed notch codes for automobile model and color.

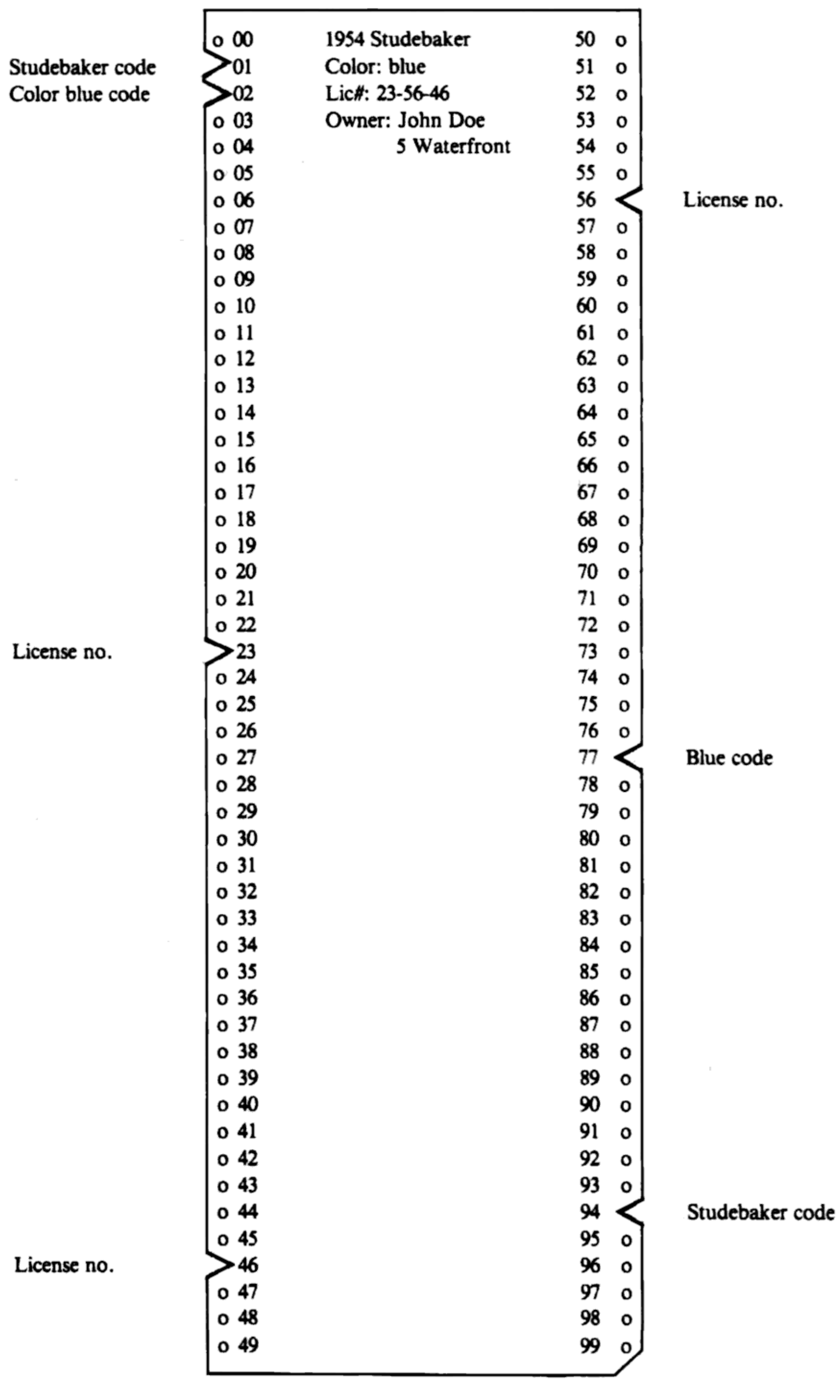


Figure 2. Edge-notched card for a blue Studebaker with license number 23-56-46.

digit numbers (72-54-81). There are one million possible license numbers.

Often, a witness to a crime remembers only one or two of the pairs of digits, so our detective devises a method that lets him quickly determine the owner of a vehicle when he knows only part of a license number.

He buys cards with 100 holes and uses the following coding technique: He numbers the holes in the card zero to 99. For each pair of digits in the license number, he makes a notch corresponding to the number in the appropriate card hole (Fig. 2). Additionally, he codes 20 models and 10 car colors as shown in Table 1.

The detective obtains a registration list from the Department of Motor Vehicles (DMV) and starts to code all the information about 200,000 automobiles in his city onto cards. One year later, just as he finishes his task, he gets a call from a client whose TRS-80 was stolen.

The client saw the get-away car (a blue Studebaker) drive off, but could read only the first and last pairs of digits from the license plate. The DMV computer is broken so the police can't search the motor vehicle registry data base until tomorrow, but the client needs his computer today.

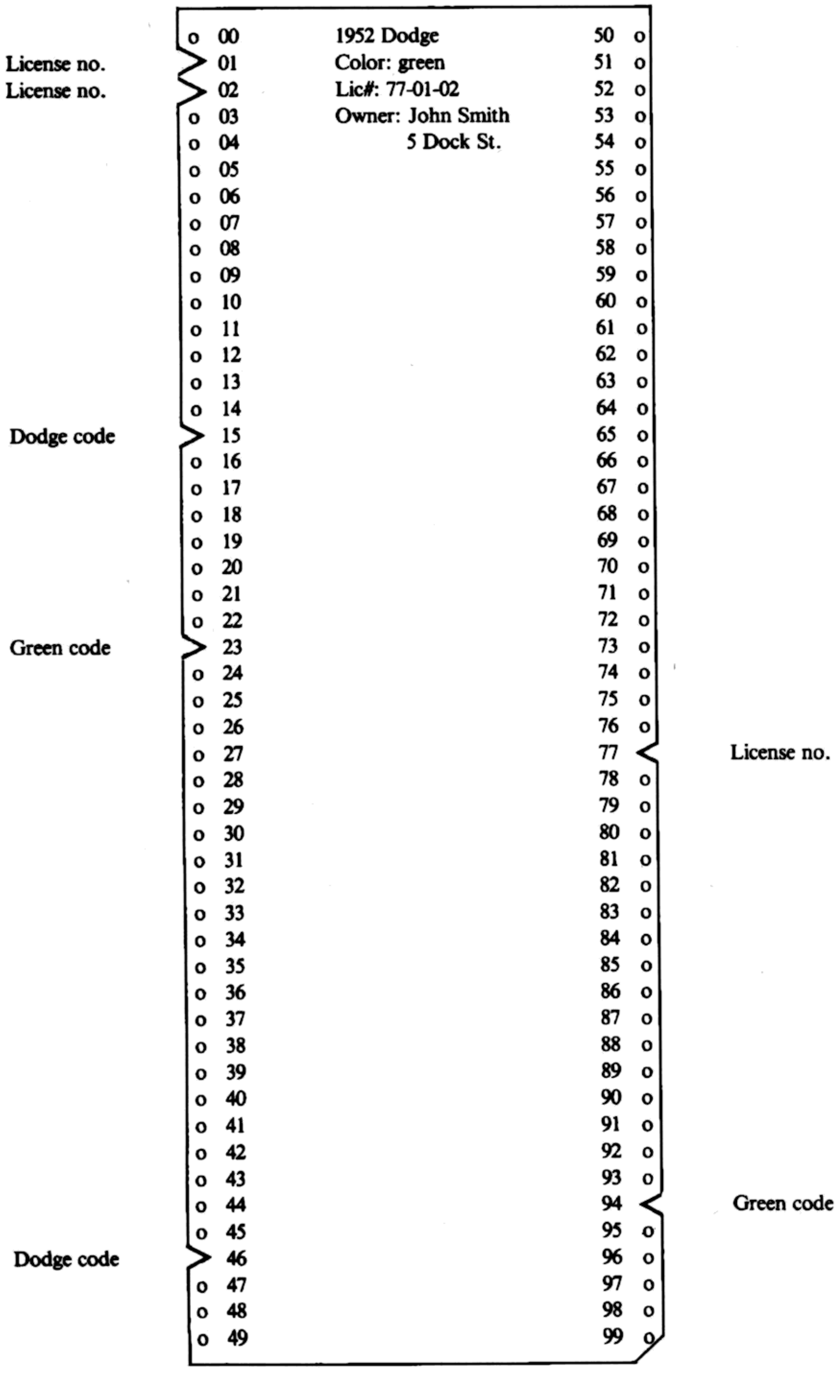


Figure 3. Edge-notched card for a green Dodge with license number 77-01-02.

The detective takes his cards, puts them in a neat deck, then places very long needles through the holes corresponding to the digit pairs 46 and 23, the color blue, and the automobile model Studebaker. Three cards fall out of a deck of 200,000 cards. The detective sequentially searches the three cards and finds that only two are Studebakers. He has narrowed his search from 200,000 to two.

Both Studebakers are blue: one with license number 46-79-23 and the other 23-46-42. The model and color codes are the same and the detective specified only two of the three possible two-digit numbers from the license (23 and 46), so the signatures of both automobile records fit the search.

But why did the third car, a green Dodge, also fall out? The code for the color green is 23 94 and the code for the model Dodge is 46 15. The green Dodge that fits the search key had license number 77-01-02. The notches in the card (23, 94, 46, 15, 01, 02, 77) corresponded to every needle placed in the deck (Fig. 3), so the card fell out along with the two Studebaker cards.

The detective used a primitive coding scheme. Elaborate methods have been developed for both edge-notch cards and bit-string computer searches. Signature screening, which you'll use to search text files, is a variation on the method the detective employed.

Tree Search

In tree searching techniques, records are arranged in some order (numerical, alphabetical, and so on). Either the records within the data file are ordered, or a separate index file contains ordered information (keys) that points to the appropriate records in a file.

You start searching anywhere in the file. If your search key doesn't match a record key, you move your search closer to the beginning or end of the file depending on whether the search key is greater or less than the key of the record just searched.

As an example of this searching technique, consider the slow-witted detective. This time he has a suspect's name (John Smith) and wants to know where the suspect lives. Since names in the telephone book are ordered (sorted alphabetically), the detective uses the following search algorithm.

He opens the telephone book to the middle and finds names beginning with the letter K. He knows that the letter S is between K and Z, so he divides the remaining pages in half and finds names beginning with R. One more division

★ FREE SHIPPING ★
WITHIN CONTINENTAL 48 STATES
ON PREPAID CASH & COD ORDERS ONLY.

MICRO IMAGES INDUSTRIES INC.

MICRO IMAGES INDUSTRIES INC.

TRY US!

TOLL FREE ORDER LINE
1-800-223-1182
ORDERS ONLY
New York Orders & Information
Call (212) 445-7124

STOP & SHOP
☐ BOOKS ☐ SOFTWARE
☐ PRINTERS ☐ RIBBONS
☐ COMPUTERS ☐ ACCESSORIES
☐ SPOOLERS ☐ MODEMS
☐ DISKETTES ☐ MONITORS

VISA — COD — MasterCard

FREE RIBBONS
Receive a FREE RIBBON when you place an order over $100.00 and mention this ad. Specify which cartridge or reload you require when ordering. Offer includes all reloads and cartridges listed except MX100, ANADEK and CENTRONIC cartridges. Offer good November 30, 1983. Limit one per order.

$124.95 DATAWRITER NEW VERSION SPECIFY MDL I or III

$129.95 COPYART II MODEL I or III SPECIFY PRINTER

$89.95 MULTIDOS SPECIFY MDL I or III SINGLE OR DOUBLE DENSITY

$139.95 LAZYWRITER MDL I, III or IV SEE LAZY OPTIONS BELOW

NEW SCRIPT 7.1 $109.95
with Mailing Label Option . . $119.95
Mailing Label Option Only . . . 27.95
See Prosoft Options Below

MAXI CRAS Model I/III **$79.95**

MAXI MGR. Model I/III **$99.95**

SUPERUTILITY + Mod I/III Ver. 3.1 $59.95
See New Book Below

DOSPLUS Ver. 3.5 Model I or III **$119.95**

MZAL Rel. III Model I or III **$84.95**

LDOS Model I or III **$114.95**

DOTWRITER 3.0 **$69.95**

SNAPP WARE **10% Off**

DOSPLUS II Model II **$199.95**

ELECTRIC WEBSTER w/corr **$129.95**

ZORLOF Model I or III **$64.95**

PROSOFT UTILITIES
TRASHMAN - Model I or III $34.95
FASTER - Model I or III 26.95
RPM - Model I or III 21.95

LAZY OPTIONS
LAZY FONT (Epson) 44.95
LAZY MERGE 39.95
LAZY DOC 49.95
LAZY Draw, Do, Tab, Calc 34.95

MAIL LIST PROGRAMS
POWER MAIL PLUS-Mdl 2/12/16 . . . . . . 129.95
POWER MAIL PLUS-Mdl I/III 129.95
POSTMAN MASS MAIL-Mdl I/III 74.95
MAXI MAIL-Model III 79.95

TALLYMASTER - Model I or III 69.95

MICROTERM - Model I or III 69.95

TZAL — Model III Tape 39.95
BASIC EDITOR - Model I/III Tape 24.95

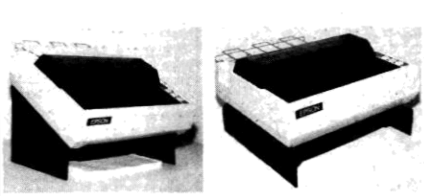
MAS 80 Acctg System
Integrated G/L, A/R, A/P & Ck Reg.
Retail $599.00 ***OUR PRICE $499.00***
Overview — $5.00 Credited on Purchase

BOOKS
How To Do It on the TRS 80 IJG 27.95
Machine Lang. Disk I/O IJG 27.95
TRSDOS 2.3 Decoded IJG 27.95
Disk + Other Mysteries IJG 20.95
Basic Decoded IJG 27.95
Custom TRS-80 IJG 27.95
Basic Faster & Better IJG 27.95
Model II/16 Visicalc W.C.Brown 16.95
Model I/III Visicalc W.C.Brown 16.95
BFBLIB or BFBDEM Disk 17.95
Inside Superutility Plus 2.2 or 3.0 . . . . . . 17.95
Electric Pencil Manual 22.95
CPM Users Guide 15.95

SPECIAL
LNW 80 Model II
With Software Bonanza!
Includes: LNW Accounting Series - G/L, A/P, A/R & Payroll, Electric Spread Sheet, Electric Pencil, Microterm, Chart-ex, CPM 2.2, DOSPLUS & LNW Basic.
RETAIL $1995 ***OUR PRICE $1695.00***

LNW SYSTEM EXPANSION II 349.95
LNW 5/8 DOUBLER 169.95

PRINTER STANDS



	SPACE AGE	NO FRILLS
MX80 No Frills	Clear 13.50	Bronze 15.95
MX80 Space Age	Clear 27.50	Bronze 29.95
Microline 82A Space Age w/slot	Clear 27.50	Bronze 29.95
Microline 83A Space Age w/slot	Clear 34.95	Bronze 39.95
NEC 8023A Space Age	Clear 27.50	Bronze 29.95
MX100 Space Age	Clear 34.95	Bronze 39.95
Multiuse - 18x12x6x¼	Clear 24.95	Bronze N/A

MICROBUFFER Parallel or Serial (Epson) . . . 139.95

GEMINI 10X 329.95

MICRO SPOOLER By Consolink 16K 179.95
64K 269.95

HAYES SMARTMODEM 300 . . . . 229.95
HAYES SMARTMODEM 1200/300 . . . . 579.95

LYNX AUTO DIAL/ANSWER MODEL I/III 239.95

RIBBONS
ZIP BOX RELOADS

	½ Doz.	Doz.
Epson MX 70/80-20 Yds.	24.00	42.00
Epson MX 100-30 Yds	30.00	52.00
NEC/Prowriter-14 Yds	21.00	36.00
Centronics 730/737/739/779 or LP-I/II/IV-16 Yds	18.00	32.00

All ZIP BOXES are individually sealed black nylon and require no rewinding. Epson Reloads also available in red, blue, brown, green & purple. Any mix allowed.

CARTRIDGES

	Each	Doz.
Epson MX70/80	7.00	70.00
Epson MX100	12.00	125.00
Prowriter 8510 & NEC 8023A	7.50	80.00
RS LP III/V	6.50	70.00
RS LP VI/VIII	6.50	65.00
RS DSY WH II or DWP 410	6.50	70.00
RS DSY WH II - Nylon	6.50	70.00
MICROLINE 80/82A/83A/92	N/A	30.00
MICROLINE 84 ½ x 40 Yds	5.50	60.00
ANADEK - 9000 Series	13.00	135.00
Diablo Hytype II-Multi Strike	6.50	65.00
Qume-Multi Strike	5.00	50.00
NEC-5500/7700-Multi Strike	7.00	70.00
Centronic 703/04/53	11.00	120.00

Note: All cartridges black only.
Minimum order 3 cartridges - any mix. For smaller quantities, add $1.50 per order. All our reloads and cartridges are manufactured by one of the oldest and most reputable ribbon manufacturers in the country.
—QUALITY GUARANTEED—

FOR THE CO CO
Sea Dragon - 32K Tape 28.95
Rearguard - 16K Tape 19.95
Diskey - 32K Disk 39.95
Eliminator - 16K Tape 19.95
Triad - 32K Tape 28.95
Earthquake - 16K Tape 19.95

Micro Images Industries Inc.
164-06 Crocheron Ave. Dept. L
Flushing, N.Y. 11358
(212) 445-7124
HOURS: Mon.-Thurs. 10AM-6PM; Fri. & Sat. 10AM-5PM

VISA — MasterCard

✓157

ORDERING INFORMATION
COD Orders add $2.00. Minimum credit card order $25.00. FREE shipping on all COD and prepaid cash orders within the continental 48 states via UPS Ground. Actual shipping and insurance charges apply on all credit card orders, approved P/O's and shipments outside the continental 48 states. Certified Ck, M/O, COD and credit cards shipped immediately. Please allow 2 weeks for personal checks. Credit card orders shipped to card address only. Prices subject to change without notice. New York State residents please add appropriate sales tax.

and search, and he finds names beginning with T.

He continues his search in the opposite direction, and after five more divide-and-search sequences he locates 25 John Smiths. The detective could have made his search more efficient if he had used information he already possessed: his knowledge that names beginning with S are closer to the back of the telephone book than to the front.

The many different tree searching techniques include binary searches, indexed sequential access methods, and so on. All require that the information you're searching for have some order so that comparisons between a search key and a record key determine in which direction the search branches (toward the beginning or end of a file).

Hash Search

In hash searching techniques, you can store the source file records in any order. The hash searching program performs a sequence of mathematical operations (hash functions) on the search key, and produces a value that points to the appropriate record.

The detective has decided that he must use high technology to keep up with the competition. He buys a Model 12 and a 12-megabyte hard disk drive, has a data-entry clerk type all the information from the telephone directory into one large file, then hires a high school student to write a data retrieval program that quickly searches the file by name, number, or address.

The student realizes that sequential search techniques are too slow and that Microsoft Basic doesn't support ISAM files. She writes a program to convert every name into a code that indicates where the program stores the record in the file.

The hash function is simple: The ASCII values of the characters comprising an individual's name are multiplied together, then divided by 200,001. The remainder of the division is the record number at which the program stores the name, address, and telephone number. Formally,

$$h(k) = (\overset{n}{\pi} a_i) \bmod 200{,}001$$

where h is the hash function, k is the key (in this case the person's name), n is the number of letters in the person's name, and a_i is the ASCII value of the i^{th} character in the person's name. π is the product $a_1 \cdot a_2 \cdots a_n$ and mod is the modulus of the quantity in parentheses (the remainder after the number in parentheses is divided by 200,001).

Because the program listings in this article are so long, we had to print them at a size smaller than usual. We apologize for any inconvenience this causes.

To buy these programs on disk, see the ordering instructions on p. 204.—Eds.

The student finishes her task and goes away for summer vacation. The detective needs to find the address of James Joyce. He sits down at his computer and types in the search key, "James Joyce". Instantly a name, telephone number, and address appear on the video display. Unfortunately, the name is Joyce James, not James Joyce. Our detective has experienced a collision—two distinct record keys hashed to the same number.

The detective is disappointed and afraid that he'll have to use the telephone book. Then he notices a message at the bottom of the screen: "Press N for next record". He presses N. The computer pauses for a brief moment and displays the name James Joyce. The student incorporated a collision-handling routine into her program.

Signature Screening

Tree and hash searching methods are appropriate for records that contain specific fields (name, address, tele-

Source Line

```
;MINDEX2/SRC...1.2
;Mar 05, 1983
;-------------------------------------------------------------------
;MINDEX INDEXING PROGRAM FOR MODEL II
;MASTER SECTION: links main program with Model II I/O routines
                 ;and general purpose routines
;COPYRIGHT 1983, SOFTSHELL CORPORATION   ALL RIGHTS RESERVED
;-------------------------------------------------------------------
PROGII  PSECT                     ;begin relocatable program section
        EXTERN  BEGIN             ;BEGIN = start of M3
START   JP      BEGIN
        LINK    'M3/REL'          ;main program
                                  ;note: the program is called M3 because it
                                  ;was originally developed on Model III
                                  ;In its current form it will work with
                                  ;Model II or III given the proper I/O
                                  ;module
        LINK    'GR/REL'          ;general programs
        LINK    'MODII/REL'       ;Model II I/O routines
        END     START
```

Program Listing 1. Assembly-language listing of Mindex2/SRC.

Source Line

```
;SEARCH2/SRC...1.2
;Mar 05, 1983
;-------------------------------------------------------------------
;SEARCH PROGRAM FOR MODEL II
;MASTER SECTION: links S3, GR, S3OPEN, S3DSP, S3OPT and MODII
;COPYRIGHT 1983, SOFTSHELL CORPORATION   ALL RIGHTS RESERVED
;-------------------------------------------------------------------
PROGII  PSECT                     ;begin relocatable program section
;-------------------------------------------------------------------
        EXTERN  BEGIN             ;BEGIN = start of S3
;-------------------------------------------------------------------
START   JP      BEGIN
        LINK    'S3/REL'          ;SEARCH program proper
                                  ;note: the program is called
                                  ;S3 because it was originally
                                  ;developed on the Model III
                                  ;In its current form it will
                                  ;work with Model II or III
                                  ;given the proper I/O module
        LINK    'GR/REL'          ;general routines
        LINK    'S3OPEN/REL'      ;SEARCH open files routine
        LINK    'S3DSP/REL'       ;SEARCH display
        LINK    'S3OPT/REL'       ;SEARCH options
        LINK    'MODII/REL'       ;Model II I/O routines
        END     START
```

Program Listing 2. Assembly-language listing of Search2/SRC:1.

Program Listing 3. Assembly-language listing of MODII/SRC.

Source Line

```
;MODII/SRC VERSION 1.2
;Jun 21, 1983
;-------------------------------------------------------------------
MODII   PSECT   ;Model II I/O routines
;-------------------------------------------------------------------
;MAP MODII I/O ROUTINES, BUFFERS AND PARAMETERS
;COPYRIGHT SOFTSHELL 1983 ALL RIGHTS RESERVED
;-------------------------------------------------------------------
MODII   PSECT                     ;begin relocatable program section
;-------------------------------------------------------------------
                                  ;public routines
                                  ;---------------------------------
        PUBLIC  BOTTOM            ;position cursor to last line of screen
        PUBLIC  CLOSE             ;close file (DE->DCB)
        PUBLIC  CURPOS            ;determine cursor pos (coor:BC)
        PUBLIC  DISBRK            ;disable break (for Model II)
        PUBLIC  DIVIDE            ;divide A into HL
        PUBLIC  DOSCMD            ;jp to TRSDOS and execute command
        PUBLIC  ERASEL            ;erase to end of line
        PUBLIC  ERASES            ;erase to end of screen
        PUBLIC  ERRDSP            ;MAP error display routine
        PUBLIC  HOME              ;position cursor to top line of screen
        PUBLIC  INIT              ;initialize file (create/open, pos to 1st rec)
                                  ;             (DE->DCB, HL->rec buf)
        PUBLIC  JP2DOS            ;jp to TRSDOS
        PUBLIC  KBCHAR            ;get char (or null) from keyboard
        PUBLIC  KBINIT            ;initialize keyboard (Mod II)
        PUBLIC  KBLIN3            ;get line from keyboard and end with 03
        PUBLIC  KBLINE            ;get line from keyboard and end with 0DH
        PUBLIC  KBWAIT            ;wait for char from keyboard
        PUBLIC  KEYIN             ;get search key(s) form keyboard
        PUBLIC  MOVELN            ;move line from screen to buffer
                                  ;HL->buf, D=no. char, BC=cursor coor
        PUBLIC  MULT              ;multiply routine
        PUBLIC  NEWLN             ;role screen and pos cur to next to last line
        PUBLIC  OPEN              ;open file (DE->DCB, HL->record buffer)
        PUBLIC  OPINIT            ;open and initialize file
                                  ;        (DE->DCB, HL->record buffer)
        PUBLIC  POSCUR            ;position cursor (coordinates in BC)
        PUBLIC  POSEOF            ;position to end of file
        PUBLIC  POSN              ;position to record in file
        PUBLIC  PRINIT            ;initialize printer
        PUBLIC  PRLINE            ;print line (HL->buf, B=no. char, C=end char)
        PUBLIC  PRSCN             ;print entire screen
        PUBLIC  PRTBLN            ;print last line of text
        PUBLIC  PRTREC            ;print current record
        PUBLIC  PTFS              ;point to file spec in cmd line
        PUBLIC  READDR            ;read file: direct access
        PUBLIC  READNX            ;read next record in file
        PUBLIC  SCRPRO            ;set scroll protect line (A=line no.)
        PUBLIC  SETILR            ;set INLRR to true last record
        PUBLIC  SETMLR            ;set MAPLLR to true last record
        PUBLIC  STMEND            ;store top of memory in MEMEND
        PUBLIC  VDCHAR            ;display char in A register
        PUBLIC  VDCLS             ;clear screen
        PUBLIC  VDINIT            ;initialize video I/O (for Model II)
        PUBLIC  VDLIN3            ;display line (HL->msg; don't print 0DH)
        PUBLIC  VDLINE            ;display line (HL->msg)
        PUBLIC  WPKBWT            ;word processor kbwait routine
        PUBLIC  WRITE             ;direct write routine
        PUBLIC  WRITNX            ;write next record to file
```

Listing 3 continued

String	He ran down the lane							
Triplets	ran	dow	own	the	lan	ane		
Hash value	181	185	101	191	83	108		
String	Jill walks up the lane							
Triplets	jil	ill	wal	alk	lks	the	lan	ane
Hash value	212	142	177	94	188	191	83	108
String	The goat ran up the lane							
Triplets	The	goa	oat	ran	the	lan	ane	
Hash value	191	212	138	181	191	83	108	

Table 2. Three strings, their corresponding triplets, and hash values.

phone number, and so on). Text, however, does not lend itself readily to such indexing, and many text searching programs still employ sequential search techniques.

In 1971, M.C. Harrison proposed a method to speed up searching large text files ("Implementation of the Substring Test by Hashing." Comm. Assoc. Computing Machinery. 14:777–779, 1971). The idea is simple: Before searching a string of text for a particular key, perform a quick screening test to determine whether the key is not present or possibly present.

If the test determines that the key isn't present in a string, you don't need to search that string. If the test is positive (the string might be present), a sequential search determines if the key is actually present in the string. I refer to this method as signature screening.

The following method is an extension of that which Harrison proposed. It's designed for use on 8-bit machines that read 256-byte disk sectors. Each record consists of a string of ASCII characters.

The search program considers every three sequential characters (triplet) of a string as attributes of the string. The program ignores triplets containing one or more spaces. For example, the string "Jack ran up the hill" has the following attributes: "Jac", "ack", "ran", "the", "hil", and "ill".

Considering only letters of the alphabet, you can have 26^3 or 17,576 possible attributes for any triplet of letters A through Z. The 17,576 possible attributes (triplets) are mapped onto a 251-bit string. A hashing function reduces the 17,576 attributes to 251 values (zero to 250).

The hash function, h, for any triplet is:

$$h(c_1,c_2,c_3) = (100a(c_1) + 10a(c_2) + a(c_3)) \bmod 251$$

where c_1, c_2, and c_3 represent the first, second, and third characters of the triplet respectively, and a represents the ASCII value of any character. Any uppercase letter is assigned the same value as the corresponding lowercase letter. The mod is the modulus of the quantity in parentheses (the remainder) after the number in parentheses is divided by 251. If any triplet has one or more blank characters in it, it has no hash value.

The word "hill" has two triplets. You want to find the hash value of the first triplet, "hil". The ASCII values of "h", "i", and "l" are 104, 105, and 108, respectively.

First multiply 104 by 100, 105 by 10, and 108 by 1. Then add the three numbers (11,558) and finally divide 11,558 by 251. The remainder, 12, is the value of the hash function for the triplet hil.

The hash function has several features that are important when indexing text. The order in which letters are arranged in a triplet determines the hash value. Triplets such as "tab" and "bat" do not hash to the same value.

Spaces, which often consume a considerable portion of text, aren't included so the program indexes only words of three or more characters. This avoids indexing many meaningless triplets. In the string "boy and girl", the program hashes and indexes only the triplets "boy", "and", "gir", and "irl". The triplets "oy ", "y a", "nd ", "d g", and " gi" aren't included. Last, the computation of the hash value is simple and quick.

The signature of a string consists of 251 bits initialized to zero. If a triplet of the string hashes to a value i, the search program sets the i^{th} bit (b_i) of the signature to 1 (equivalent to cutting a notch in the i^{th} hole of a notched-edge card). Signatures of file records are called record signatures while those of search keys are called key signatures.

It's easy to find a single key in a large text file. First, the search program loads the record signatures of the text file into memory (or as much as will fit at one time). Second, the program determines the signature of the key. Then it sequentially compares each record signature to the key signature.

If for every b_i equal to 1 in the key signature, a corresponding b_i is equal to 1 in the record signature, the screening test is positive, and the program brings the record into memory and searches for the key. If for any i, b_i equals 1 in the key signature and b_i doesn't equal 1 in the record signature, the key isn't present and you don't need to search the record.

This is the same as putting needles through holes in edge-notched cards. Cards that don't fall out when you lift the needles lack at least one of the attributes you're searching for and you can bypass them.

As a concrete example, take a text file with only three strings: "He ran down the lane", "Jill walks up the lane", and "The goat ran up the lane". See Table 2 for all triplets and the corresponding hash values for the three strings.

Suppose you want to search the file for all strings that contain the name "Jill". The key signature for the name "Jill" (Fig. 4) has bits 212 and 142 set to 1; all the other bits are zero.

Comparing the signature of the string "He ran down the lane" bit for bit with the key signature, you see that b_{212} of the key signature equals 1 while b_{212} of the record signature equals zero. The screening test is negative and you don't need to search the actual string.

Comparing the signature of the string "Jill walks up the lane" with that of the key signature, you see that every bit equal to 1 in the key signature is also equal to 1 in the record signature. Read the record from the disk, search it sequentially, and you find the name "Jill" present.

Comparing the signature of the string "The goat ran up the lane" with that of the key signature, you again find that every bit equal to 1 in the key signature also equals 1 in the record signature. You read the corresponding record from the file, search it sequentially, and determine that the key isn't present (a collision has occurred). If you had more records in the file, you could continue the search in the same manner.

Searching for a string that contains two or more keys is simple. The program logically ORs the key signatures

Continued on p. 182

Assembly Language Shortcuts—Part II

by Bob Bowker

This month's shortcuts emphasize efficient use of the stack and include a reassuring perspective on the mysteries of algorithms.

```
CONDITION 1      CONDITION 2                          CONDITION 20
LD BC,0123H      LD BC,0254H                          LD BC,2017H
LD (BUF),BC      LD (BUF),BC    ==========>>>         LD (BUF),BC
CALL ROUTINE     CALL ROUTINE                         CALL ROUTINE
RET              RET                                  RET


ROUTINE LD  HL,(BUF)
        .
        .
        .
        RET
```

Program Listing 1. Stack value pass.

```
CONDITION 1      CONDITION 2                          CONDITION 20
POP HL           POP HL                               POP HL
CALL ROUTINE     CALL ROUTINE   ==========>>>         CALL ROUTINE
DEFB 32H         DEFB 54H                             DEFB 17H
DEFB 01H         DEFB 02H                             DEFB 20H


ROUTINE EX  (SP),HL
        .
        .
        .
        RET
```

Program Listing 2. Byte-efficient Stack value pass.

In my last article on Assembly language programming (*80 Micro*, June 1983, p. 173), I discussed some techniques on reducing memory requirements. This month, I'll consider the stack, a last in/first out storage area used by the Z80 to save both addresses and values as it jumps around in a program. The stack is available to the user, too; you can store addresses and values on the stack with the Push command, and reclaim them with Pop.

When you call a subroutine, the computer automatically stores the return address on the stack, so that when it encounters a return instruction, the last address on the stack tells the program where to return to.

If your subroutine has left an extra address on the stack, when it encounters the return, the program returns to the extra address, not to the proper place. Similarly, if too many addresses are popped off the stack, the address to which the program returns will be wrong.

What's Wrong is Right

Sometimes you can misuse the stack to advantage. Suppose that you have a program with 20 possible situations,

The Key Box

Model I and III
16K RAM
Editor/Assembler

each of which must pass a unique value to a common Write routine. One possibility is Program Listing 1. There is a way to use the stack to pass the value, and, in this example, save many bytes. See Program Listing 2. In each of the 20 cases here, the POP HL instruction puts the main return address into the HL register pair, and the call routine puts the address of the first DEFB onto the stack, even though we have no intention of returning here. At routine, the 1-byte instruction EX (SP),HL puts the main return address on the stack, and the HL register pair points to byte 1 of the value to be passed. When the program encounters the return instruction at the end of routine, it goes back to the original code that called the condition and not to one of the DEFB lines.

```
00100 57          CONV  LD D,A       ;Store for now
00100 E6F0              AND 0F0H     ;Dump bot 4 bits
00120 0F                RRCA         ;Do the rotate trick
00130 0F                RRCA
00140 0F                RRCA
00150 0F                RRCA
00160 CDxxxx            CALL SUB     ;Call the subroutine
00170 57                LDA,D        ;Recall the number
00180 E60F              AND 0FH      ;Dump top 4 bits
00190 C630        SUB   ADD A,30H    ;Make it ASCII
00200 FE3A              CP 3AH       ;Allow for HEX
                                     digits
00210 3802              JR C,ONE     ;A thru F
00220 C607              ADD A,07H
00230 CDxxxx      ONE   CALL 003BH   ;Output the byte
00240 C9                RET
```

Program Listing 3. ASCII converter.

```
00100 01FFFF DELAY LD BC,0FFFFH ;Delay length
00110 3Axxxx       LD A,(FLAG)  ;Get flag
00120 F5           PUSH AF      ;Save the flag
00130 CB47         BIT 0,A      ;Check bit 0
00140 C46000       CALL NZ,60H  ;Wait if it's on
00150 F1           POP AF       ;Restore orig (FLAG)
00160 F5           PUSH AF      ;...and store again
00170 01FFFF       LD BC,0FFFFH ;Delay length
00180 BB4F         BIT 1,A      ;Check bit 1
00190 C46000       CALL NZ,60H  ;Wait if it's on
00200 F1           POP AF       ;Restore orig (FLAG)
00210 01FFFF       LD BC,0FFFFH ;Delay length
00220 CD6000       CALL 60H     ;Standard wait
00230 C9           RET          ;Job's done
```

Program Listing 4. Three-pass routine.

```
00100 010000 DELAY LD BC,0000H     ;Delay count
00110 3Axxxx       LD A,(FLAG)     ;Get flags
00120 CB47         BIT 0,A         ;Check bit 0
00130 C4xxxx       CALL NZ,SUB1    ;Delay if on
00140 CB4F         BIT 1,A         ;Check bit 1
00150 C4xxxx       CALL NZ,SUB1    ;Delay if on
00160 F5     SUB1  PUSH AF         ;Save flags
00170 CD6000       CALL 0060H      ;...and delay
00180 F1           POP AF          ;Restore flags
00190 C9           RET
```

Program Listing 5. Single subroutine option.

How to Call Yourself

Shortcuts are often needed when the number of bytes available for the task is limited; if you have to fit a program in a 1K EPROM, that byte 1,025 has got to go.

I am also motivated by laziness: I don't like to type in the same code twice if I can help it. Duplicating several lines of code is a waste of both typing time and assembler space, so I came up with the subroutine in Program Listing 3 to convert a hex digit into 2 ASCII bytes to be displayed. This routine uses one return instruction to handle both passes through its code. Line 160 calls the subroutine; when the program encounters the return in line 240, it branches back to line 170 to the address on the stack and continues on until it hits line 240 again. This time, the return forces a branch back to the main calling code since that address is now on the stack.

There's no practical limit to the number of layers you can create using the one return instruction. This next example uses up to three passes through the same section of code, at various times, according to two switches. You could use it to allow for differences in how long you must wait for a disk drive motor to get up to speed. The first flag can indicate whether your speed-up modification is on, and the second flag can indicate that a full second is required for that drive to get to 300 revolutions per minute (rpm), as opposed to the half second required by newer drives. The original version is in Program Listing 4. The routine checks two flags and calls wait once for each flag that's on; then it calls a single wait regardless of the flags. Compare it to the version in Program Listing 5. The program uses a single subroutine and its return several times. Each time the return is encountered, the program branches to the last address on the stack. If bit 0 of (FLAG) is on, the return forces a return to line 140, and to line 160 if bit 1 is on. If neither is on, the return branches back to the code that called the delay.

The program initially sets BC register pair to 0000 hexadecimal (hex) instead of 0FFFF hex since the wait routine 0060 hex returns it to 0000 hex every time it's called; thus BC is set up for the next pass. That one extra count in the delay is only 14.65 microseconds, which in most cases won't hurt. The bonus in using the second routine is that 8 bytes are saved—over 25 percent!

End of the Line

The word algorithm is not only ugly, it's scary. That concept has probably kept more people away from Assembly-language programming than T-Bug ever did. An algorithm is nothing more than a way to do something.

```
3Axxxx  LINE    LD A,(4020H)    :Get CSRPOS LSB
CB67            BIT 4,A         ;Bit 4 on?
C0              RET NZ          ;Go back if yes
CB6F            BIT 5,A         ;Bit 5 on?
C0              RET NZ          ;Go back if yes
CDxxxx          CALL NULINE     ;It's a new line
C9              RET             ;Now go back
```

Program Listing 6. Algorithm check.

For example, once I needed a way to tell if the cursor was positioned at the start of a line on the screen. At one point I considered making a table of the 16 start-of-line addresses and doing 16 compares every time.

Then I found a blank page of Radio Shack's $1.95 Video Display Worksheet paper left over from Level I days, and wrote the 16 addresses in a column down the left side. I converted the addresses to binary and wrote them down, too. In every case both bit 4 and bit 5 were off; that turned out to be an absolute test. If either bit 4 or bit 5 of the least significant byte (LSB) of the address of the cursor were on, the cursor was not at the start of a line on the screen. I had an algorithm (see Program Listing 6). This routine checks bits 4 and 5; if either is on, the cursor is not at the start of a line, so it returns. If both are off, the program calls the subroutine Nuline to do whatever, followed by the main return.

> *"Don't be frightened off by jargon like algorithm; if what you're looking for, or what you've found, is a shortcut, call it that."*

Don't be frightened off by jargon like algorithm; if what you're looking for, or what you've found, is a shortcut, call it that.

To Vector or Not to Vector

Lowercase modifications on the Model I are pretty common now, but not all users have them. This poses a problem if you're going to sell, or otherwise distribute, a program to others; you have to make allowances for the unmodified computers. Also, if your program uses the printer in any way, you should allow for those who don't have one. There are few things more frustrating than to accidentally hit the P key, and sit helplessly as the computer locks up in a ROM loop looking for a printer that isn't there.

Vector is more jargon; it means jump, so let's call it a jump. That's also an easy way out of the problem: Place a jump or two at the start of your program, that can be changed by any user to allow for printers and lowercase.

For example, I wrote a program that calls a conversion subroutine just before printing every character on the screen; if the character were lowercase, it would be converted to uppercase. Before displaying the character in A on the screen, I would call LC2UC.

My program also jumped through a subroutine on its way to the line printer; to print the character in A, I would call print.

```
00100 C3xxxx LC2UC JP CONV ;Go convert
      LC to UC
00110 C3xxxx PRINT JP LPR ;Go print out
```

These were the first two lines of the program, which loaded at 5200 hex. I included instructions to zap a return into line 100 (a C9 at 5200 hex) if the user had a lowercase mod; then, every time LC2UC was called, the program found a return before it could convert anything. Similarly, the program could zap a return into line 110 (a C9 at 5202 hex) if the user had no printer; this effectively disables the print command in the program.

Was That a I or a III?

When it first came out, the Model III was advertised as able to run all Model I software. As it turns out, that just isn't so; the culprits are the new ROMs in the III, which are different from the I's.

For most people, this is not important. However, if you want to sell a program that works on both, you have a decision to make: Should you sell separate versions, or try to write one version that works on both models? If you choose to do the latter, you've still got problems. You must either make sure the program uses only those routines which are common to both models, or you must somehow have your program modify itself based on the host model.

The last choice is not difficult. You can tell whether the computer running your program is a III or a I by looking at the byte at 0054 hex in the ROM; if it's a 01 hex, you're on a Model I, and if it's an EB hex you're on a III. There are probably other addresses you can use, although this is the only reliable one I know.

"Should you sell separate versions, or try to write one version that works on both models?"

Start your program after the jumps with a check of 0054 hex, and deal with the result accordingly. For instance, if you find that your computer is a III, you may want to disable your own LC vector right away by putting a return there, and store a 00 hex at 4019 hex to default the III to lowercase. If any of the DOS calls you intend to use are different in Model III versions of the operating system, you can alter them now. For example, assume that a DOS call in Model I is at 4290 hex, and in the Model III is at 442B hex.

```
01320 CD9042 HERE CALL 4290H ;Model I
      CALL
```

If your check finds that you're on a III, include the following lines before starting the main program:

```
65000 212B44 LD HL,442BH ;Model III address
65010 22xxxx LD (HERE + 1),HL
```

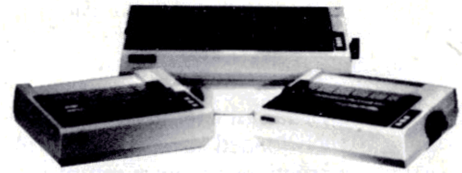
When line 1320 is encountered, the proper Model III address will be called. ■

Robert Bowker is a free-lance television director. He can be reached at 11360 Sunset Blvd., Los Angeles, CA 90049.

✓ See List of Advertisers on Page 291

DISPLAYED VIDEO IS DRIVING DOWN PRICES ON DISKS!

FREE SHIPPING!



TEC'S NEW HALF-HIGH 40 TRACK DISK DRIVE W/CASE & POWER SUPPLY AT AN INCREDIBLE LOW PRICE!

$195.00!!!!

NEW!
TRUE HALF HEIGHT DRIVES!
1 5/8 INCH NOT 2 INCH OR 2 7/16 INCH! DIRECT DRIVE!
NO DRIVE BELT! 3MS TRACK TO TRACK!

"YOU CAN BUY THE REST BUT WHY NOT BUY ONE OF THE BEST!"

TEAC's NEW SLIMLINE 40 TRACK W/CASE & POWER SUPPLY **$235.00**
TANDON 40 TRACK TM-100-1 W/CASE & POWER SUPPLY **$220.00**
(DOUBLE SIDED 40 TRACK DRIVES ADD $75.00)
SHUGART NEW SLIMLINE DOUBLE SIDED 40 TRACK W/CASE & POWER SUPPLY **$299.00**
SHUGART NEW SLIMLINE DOUBLE SIDED 80 TRACK W/CASE & POWER SUPPLY **$379.00**
TANDON SINGLE SIDED SLIMLINE 8" DISK DRIVE W/DUAL CASE & POWER SUPPLY **$545.00**
TANDON DOUBLE SIDED SLIMLINE 8" DISK DRIVE W/DUAL CASE & POWER SUPPLY **$625.00**
TWO DRIVE CASES AVAILABLE AT VARIED PRICES
TWO DRIVE 5 1/4" CABLE (FOR MOST COMPUTERS)..... **$23.99** WITH GOLD PLATED CONNECTORS

DV'S COLOR COMPUTER 1st DRIVE ONLY **$369.00**
ONE YEAR WARRANTY ON TEAC & SHUGART DRIVES/180 DAYS ON TEC & TANDON
HAYES SMART MODEM 300 BAUD .... **$225.00** 1200 BAUD..... **$525.00**
MODEL I DOUBLE DENSITY BOARD **$89.00**

PRINTER PRICES

WE'RE NOT GOING OT MAKE YOU CALL FOR PRICES — HERE THEY ARE IN BLACK AND WHITE!!!

EPSON

MX80 FT W/GRAPHTRAX PLUS $459.00
MX100 W/GRAPHTRAX PLUS $659.00
RX80 W/GRAPHTRAX PLUS $389.00
FX80 W/GRAPHTRAX PLUS $569.00
FX100 W/GRAPHTRAX PLUS $779.00

STAR

GEMINI 10 .... $349.00
GEMINI 15 .... $479.00
CITOH PROWRITER 8510 .... $395.00
STARWRITER F10 .... $1169.00
DWP 210 .... $649.00

PRINTER CABLE 10' LONG W/GOLD PLATED CONNECTORS STARTING AT $25.99
DAISY WHEEL PRINTERS AVAILABLE AT VARIED PRICES

Visit our two retail locations at:
886 Ecorse Road
Ypsilanti, MI 48197
(313) 426-5086/(313) 482-4424
or
111 Marshall Street
Litchfield, MI 49252
(517) 542-3280
(517) 542-3939
(517) 542-3947

WE CARRY TRS80, LNW, EPSON & FRANKLIN COMPUTERS

DISPLAYED VIDEO

FREE SHIPPING!!

IMMEDIATE DELIVERY
180 Days Parts and Labor Warranty
DEALER INQUIRIES INVITED
Free Shipping in the U.S.
48 Contiguous States

TO ORDER: Call (313) 426-5086 or (313) 482-4424 or (517) 542-3280
(517) 542-3939 (517) 542-3947
OR WRITE: DISPLAYED VIDEO 111 MARSHALL ST., LITCHFIELD, MI 49252

*TRS-80 is a trademark of the Tandy Corporation

Prices & Specifications subject to change without notice

CP/M III Ways

by Terry Kepner

If you're looking for CP/M compatibility on your Model III, choose one of these boards and get access to hundreds of great programs.

★★★★

CP/M Mapper III Board
Omikron Systems
1127 Hearst Ave.
Berkeley, CA 94702
$199

★★★★

VID-80 Board
Holmes Engineering Inc.
5175 Green Pine Drive
Murray, UT 84107
$279

★★★★

Shuffleboard III
Memory Merchant
14666 Doolittle Drive
San Leandro, CA 94577
$299

CP/M, Control Program for Microprocessors, was the first comprehensive in-memory disk operating system for 8080 and Z80 based microcomputers. And now it's available to those who own Model IIIs.

The three Model III CP/M conversion boards included in this review are the Mapper III from Omikron, the VID-80 from Holmes Engineering, and the Shuffleboard III from Memory Merchant.

All three boards offer useful modifications, but their features do differ. Shuffleboard is the only one that lets you perform single-drive file copies, while Mapper III alone lets you transfer files and programs from TRSDOS disks to CP/M disks.

The Holmes board stands apart in that it modifies your Model III's ROM to use an 80-character by 24-line display in standard TRSDOS environments as well as the CP/M environment.

While all three boards provide CP/M capability, you should consider their features and choose the board best suited for you.

Each modification is similar in concept: You remove the Z80 CPU and put it on the CP/M board, then plug the CP/M board into the Z80 socket. The computer now operates as either a standard Model III or a Z80 CP/M computer.

Mapper III

Omikron sells several modification boards for the Models I and III, including four versions of the CP/M board (a 48K and a 64K RAM version for each computer), 8-inch drive support for the Models I and III, and a 24-line by 80-character video display modification for the Model III.

I'll describe the Model III 64K CP/M board, without the 8-inch drive board or the 80 by 24 display modification. The Mapper III is quite small, measuring 4 inches by 6 inches (see Photo 1). Centered on the right side of the board are the socket for your computer's Z80 chip and the pins for plugging the board into your computer.

The six pages of typeset installation instructions are clear, with many pictures to clarify the procedure. The guide is a professionally prepared pamphlet, the best hardware installation guide I've ever seen.

The installation instructions are easy to follow. You unplug the computer, then locate and remove the case screws. Lift off the cover, locate and remove the Radio Frequency Interference (RFI) shield screws, then remove the RFI shield.

Remove the Z80 chip (be very careful—bent pins aren't a total disaster, but make the installation difficult), plug it into the CP/M board (observe correct orientation of the chip in the socket), and plug the board into the computer Z80 socket. Finally, reassemble the computer. The entire installation takes about half an hour of continuous work.

Now I must fault the documentation. After reading the installation guide, I looked for instructions on turning on the system and testing the board's performance, but there aren't any.

The owner's manual (15 photocopied pages of mixed typesetting and printing) details the CP/M system, describing video, keyboard, RS-232, and printer drivers, plus other important information about the Omikron version of CP/M. But it doesn't contain instructions on turning on the system in TRSDOS, CP/M, or Level II Basic.

I've since discovered that Omikron is still writing the user's manual. By the time you read this review, the guide will be finished and included with the other manuals.

When I turned my system on, strange characters filled the screen; I got no drive response and other peculiar reactions. I took my computer apart, removed the CP/M board, then replaced it and reassembled the computer. This time everything worked the way it should and the drive light came on.

The video screen displays the Omikron sign-on message, with two operation choices listed, C and T. Pressing the C key causes the Model III to boot up the CP/M disk in drive zero. Pressing T boots a TRSDOS disk. If you want Level III Basic, press T and the break key. TRSDOS operation remains unchanged.

Omikron designed the video driver to emulate the Soroc IQ120 terminal, so any CP/M programs compatible with

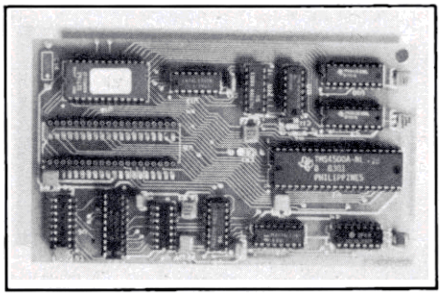


Photo 1. Mapper III.

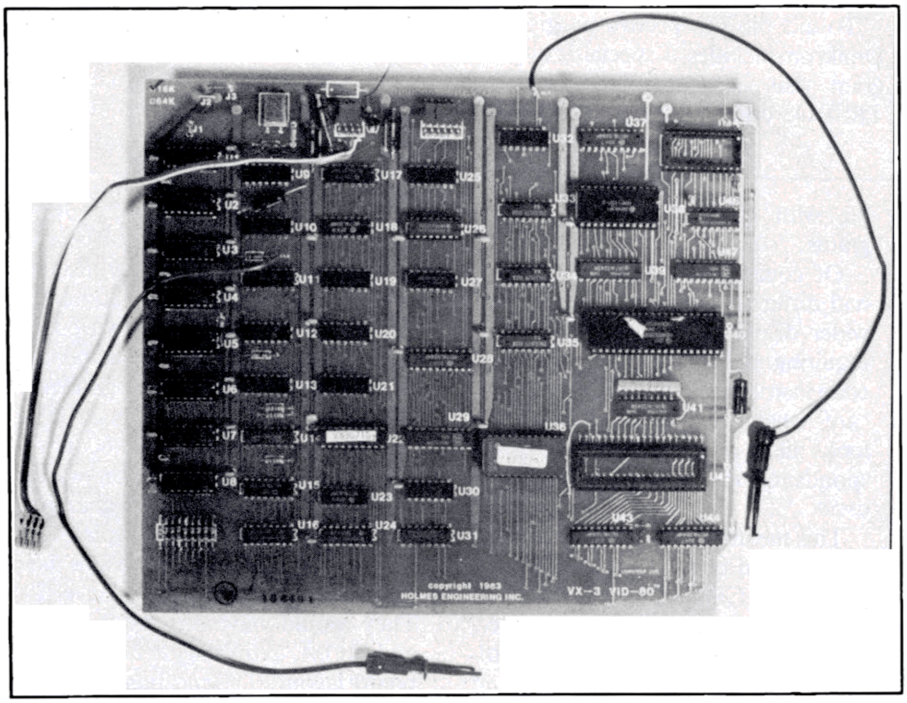


Photo 2. VID-80.

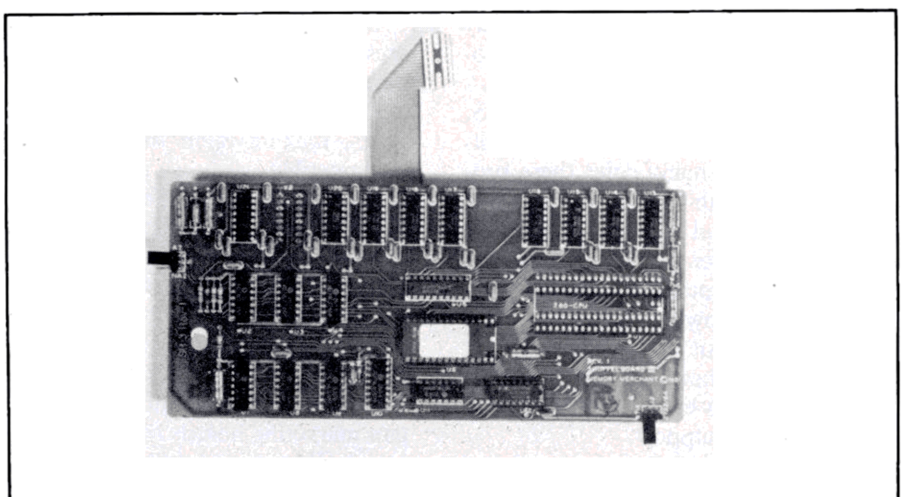
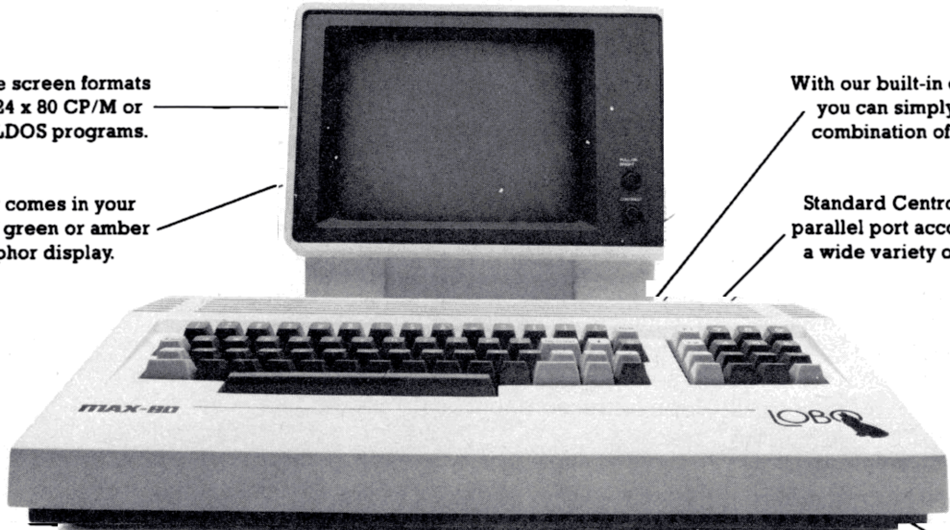


Photo 3. Shuffleboard III.

✓ See List of Advertisers on Page 291

THE COMPUTER TANDY SHOULD HAVE BUILT.



Software-selectable screen formats let you run either 24 x 80 CP/M or 16 x 32 and 16 x 64 LDOS programs.

Monitor comes in your choice of green or amber phosphor display.

With our built-in disk interfaces, you can simply plug in any combination of peripherals.

Standard Centronics-type parallel port accommodates a wide variety of printers.

Our 5MHz Z-80B processor runs your programs 25% faster than TRS-80 Model 4.

Rugged construction includes heavy duty case and full-stroke sculptured keyboard.

Available storage includes Lobo's 5-1/4" floppy, 8" floppy and Winchester hard disks.

All Lobo hardware products carry a limited one-year parts and labor warranty.

AT THE PRICE TANDY SHOULD HAVE CHARGED.

The bottom line is this. For far less than the price of a TRS-80® Mod 4, you can own a lot more computer.

A computer that's 25% faster.

That supports both the CP/M® Plus and LDOS™ operating systems.

A computer that features a price tag of only $945 for the MAX-80® processor. And that expands with your needs without breaking your budget.

A FLEXIBLE SYSTEM AT A FLEXIBLE PRICE

When you put it all together—an entire system, including a dual 5-1/4" floppy drive subsystem (320 Kb of disk storage), monitor, CP/M and LDOS—totals up to just $1599.

Totally remarkable.

But that's not all. Because this basic MAX-80 system is incredibly flexible. Its dual operating systems run far more software than any other computer. (Including our specially-discounted Perfect-Calc™ and Perfect-Writer™ software packages.)

And since our MAX-80 features two RS-232C serial ports, you can easily add both printer and modem without switching back and forth.

SUPPORTS A WIDE VARIETY OF PERIPHERALS

What's more, Lobo offers almost any peripheral you might want to add to your MAX-80. At very special prices.

Like $995 for a 5Mb Winchester hard disk, the ultimate in fast, accurate data storage.

Or a whole line of affordable dot matrix and letter-quality printers. Or an 8" floppy drive. Or even a MAX-80 local area network.

EVERY PIECE OF HARDWARE BACKED BY OUR UNBEATABLE WARRANTY

Because every part of the MAX-80 system is so rugged and reliable, we go beyond industry-standard 90 day warranties. We back every piece of Lobo hardware with a full year warranty.

So now, you can run virtually all your TRS-80 programs faster and cheaper.

Just give us your Visa or Master Card number. We'll rush you the MAX-80. Try it out. Then, if for any reason you change your mind, return it within 30 days. We'll return all your money.

So call us. Because owning a MAX-80 costs you $945. But finding out about it costs you nothing.

1-800-235-1245
(1-800-322-6103 in California)

LOBO SYSTEMS, INC.
358 South Fairview Avenue,
Goleta, California 93117

✓535

✓ See List of Advertisers on Page 291

the Soroc terminal should work without alteration. The cursor addressing routine is the same system used by the Osborne accounting packages, WordStar, and Wordmaster.

If you have 8-inch disk drive capability, Omikron's version of CP/M automatically supports it, assuming that drives A and B are 5¼-inch, and C and D are 8-inch. Since much software is sold on 8-inch disks, Omikron provides an option that lets you use control/C to warm-boot the system and make the 8-inch drives (physically located at positions C and D) the logical drives A and B.

If you have a special drive configuration, you can use the Options utility to preset the system to match your needs. The Options program controls other factors, including video graphics enable, blinking cursor, printer line counter and line feeds, RS-232 and keyboard swapping (lets you use a remote terminal to control the system), and interrupt disable.

At the moment Omikron CP/M supports only 5¼-inch single- and double-density disk drives. As soon as they complete their 8-inch double-density board, it will support 8-inch single- and double-density disk drives. In the meantime, Omikron CP/M reads 17 disk formats, including Osborne, IBM PC, Cromemco, Xerox, DEC, and Superbrain.

"...Omikron includes a Microsoft Basic disk with their CP/M system..."

To aid the prospective programmer, Omikron includes a special utility program that lets you read Model III TRSDOS disks, and copy files and programs from TRSDOS to CP/M. These programs might or might not work under CP/M.

Machine-language programs that use their own input/output (I/O) routines won't work under CP/M, since the peripherals have all been remapped to new locations. Basic programs should work with MBasic, unless they have machine-language PEEKs and POKEs.

To help the Basic programmer, Omikron includes a Microsoft Basic (MBasic) disk with their CP/M system, so you can immediately begin using your system to create standard Basic programs (most Model III Basic commands are used in MBasic). This is important since CP/M doesn't include a Basic programming system, although some versions of CP/M include a public domain version of Basic called EBasic.

Finally, Omikron supplies a set of new CP/M and MBasic manuals that are truly superior to any I've seen on

The CP/M Story

The birth of CP/M resulted from a need for a microcomputer disk operating system back when micros were only the toys of computer hobbyists and professionals. A few pioneers in the industry had managed to build computer systems that included a CPU, had the capability of running a video display, and would let you input data and instructions from a keyboard.

Someone had even figured out how to connect disk drives to the machine and transfer data between the two. One programmer, at the request of a business friend, took one such system and wrote a program that supervised the operation of the computer, monitor, keyboard, printer, and disk drives. This original DOS design was for an 8080 Altair.

The program was so successful at controlling the computer without lots of work and attention from the user that other people began asking for copies. Finally, Digital Research Inc. began selling the DOS and it rapidly spread across the country.

It was so popular because it was designed to be device independent. The CP/M system used set addresses for sending and receiving information to and from the various peripherals. A new computer merely had to honor those addresses to be compatible with CP/M and CP/M programs.

Of course, each model computer had different peripherals, some with wider displays or better keyboards, requiring different driver machine-code instructions, but CP/M doesn't care what the actual driver code looks like, only that the driver responds to the proper CP/M data address.

This means that a program written on a CP/M computer, as long as it honors the CP/M I/O calls, runs on almost any other CP/M computer. If your computer is from a new company, or even if you designed and built it yourself, you still have many CP/M programs available for it. Owners of non-CP/M computers have to wait for programmers to buy the computers and start writing programs for them.

As the oldest available DOS, CP/M has the most programs written for it. In many cases these programs are public domain, free for you to use and change. This public software is often of very high quality.

Other computer companies marketed their own DOSes for their computers, but CP/M had already become a standard. And people didn't want to buy a computer without software support.

When Tandy released the original Model I, they didn't design it to be a disk-based computer so they ignored CP/M. When Level II came out, it too ignored the possibility of CP/M because CP/M requires the use of low memory for its driver addresses and high memory for its own code. The Radio Shack uses all low memory addresses for itself and high memory for programs.

To allow for both TRS-80 Basic and future CP/M expansion would have required that Tandy abandon their Level I customers or replace the entire CPU Level I board when they upgraded the machines to Level II. They would have to switch RAM and ROM, requiring a new circuit board layout. The mechanical problems of redesigning the computer and replacing the units in the field were too costly.

CP/M requires using the low addresses of RAM as its tie points to the peripherals. The Models I and III use the low addresses for other purposes, making them fundamentally incompatible.

For a while, a doctored version of CP/M was available for the Models I and III, but it experienced compatibility problems with much of the CP/M software. The software expected the peripheral connection addresses to be in low memory.

Now several companies have begun marketing boards that let Model I and III owners convert to CP/M operation while still maintaining compatibility with TRSDOS. ■

CP/M. The CP/M manual is 250 pages long in 8- by 11-inch format.

The MBasic manual was written in 5- by 7-inch format, but has been reproduced on 8- by 11-inch paper, with two 5- by 7-inch pages on one 8- by 11-inch piece of paper. It's a little confusing at first, but usable. Both manuals are typeset and punched for insertion in a three-ring binder.

VID-80

The Holmes CP/M system is a much larger board than the Mapper III, measuring 8 inches by 11 inches (see Photo 2). As with the Omikron board, you remove the Z80 CPU from the Model III and plug it into the Holmes board. However, you also remove the character generator ROM, the power supply connector, and the video connector from the old CPU board, and plug them into the Holmes board.

Then you attach two wires from the VID-80 to the Model III with small clamps, and plug the Holmes video controller into the slot on the old CPU board. These modifications make it possible for the Model III to run CP/M, and alter the video to use 80 characters by 24 lines as the standard display.

In fact, if you are only interested in an 80 by 24 display, you can buy the VID-80 board alone, and forget CP/M. CP/M is not required for the 80 by 24 display, and you can purchase it separately for $120 extra.

This new display size has an important implication for CP/M operation: Some software on the market requires an 80 by 24 display. A display size other than that can cause problems when you try to use software designed for the larger display. Fortunately, much of the CP/M software doesn't have a display size requirement, so you can use a standard 64 by 16 screen display.

The documentation is a 34-page 8½- by 11-inch booklet that clearly explains every step for dismantling your computer, installing the Holmes board, and putting everything back together. It's not as slickly produced as the Omikron instructions, but is adequate to the task.

The instructions on CP/M are divided into two distinct sections. The first is part of the installation manual, and provides technical information about the CP/M board and the Holmes CP/M Basic Input/Output System (BIOS) information. It also includes brief instructions on turning on and using your Holmes modified computer and an introduction to the CP/M operating system.

The second part of the CP/M instructions is in the form of the Sybex book about CP/M by Rodnay Zaks, *The CP/M Handbook with MP/M.* Rather than plowing through the CP/M 2.2 Digital Research manuals, which tend to be confusing and hard to read, you can study a well-written tutorial on the CP/M system, with indexes and appendixes to help guide you through it.

The theory of operation for the Holmes board is simple: When you turn on the computer or press reset, the Holmes board bootstrap ROM checks to see if you're holding down the 6 key. If you are, the bootstrap ROM relinquishes control to the Model III ROM and your computer acts as though it's unmodified.

If you're not holding down the 6 key, the bootstrap ROM copies the Model III ROM to a 16K bank of RAM on the Holmes board, then patches the Model

III ROM in RAM to use the new 80 by 24 display. Every point where the ROM would normally jump to the Radio Shack video driver changes to use the Holmes video driver.

The locations normally assigned to the video RAM (3C00 to 3FFF hexadecimal (hex)) aren't used for video; the system ignores them. The new video RAM location is the uppermost 2K of RAM in your computer (F800 to FFFF hex).

Normally this means that you wouldn't have a full 64K of user RAM, but the Holmes board uses a bank-switching technique to flip the video RAM into place when the computer wants to access it, and to flip normal RAM in place the rest of the time. Thus your high memory Model III drivers are preserved for your use without alteration.

This method works because the Z80 can do only one thing at a time: When it's working on video RAM it can't use the high memory drivers or information, and vice versa. This is an interesting and efficient method of handling the video memory problem.

Unfortunately, by moving the video RAM to a new location, any programs that bypass the video's device control blocks by going directly to video locations in the range 3C00 to 3FFF hex won't work in the 80 by 24 display mode, so you have to remember to set the display to 64 by 16 when using those programs.

When you first install the Holmes board, you have to adjust the video display circuit board in the Model III to get all 24 lines on the screen. This takes only a few moments, and it's kind of fun watching the letters on the display scrunch down and new lines appear from the bottom.

"The CP/M mode is quite transparent. . . "

The new 80 by 24 display requires that the individual characters be smaller, in order for all of them to fit on the screen. That's a small price to pay for almost doubling (from 1,024 to 1,920) the amount of information on your screen—an 80 percent increase.

Another disadvantage is the appearance of snow on the video whenever you update the display, much like the snow on a normal Model III during high-speed display access (usually apparent during machine-language games). Fortunately, Holmes Engineering has a modification that makes the display clear and precise. Newer boards come with the modification.

This approach, moving ROM to RAM and modifying it to use the 80 by 24 display, lets you use almost any Model III DOS with the 80 by 24 display. For example, TRSDOS works quite well in the 80 by 24 mode, with only the DIR command requiring a patch [PATCH *6 (ADD=5AFA,FIND=3F,CHG=OF)] to operate correctly in both display modes. Similarly DOSPLUS 3.4, 3.5, and 4.x, NEWDOS80, LDOS, and MULTIDOS work with little or no alterations in both modes. They require no special software drivers.

One other feature, reverse video, is accessible from either Model III mode or CP/M mode.

The CP/M mode is quite transparent to the operator. Memory management on the Model III in CP/M mode gives you 62K of RAM (the other 2K is for disk I/O buffers and other miscellaneous data involved with the disk drives).

The Holmes board uses a 4K EPROM for keyboard, video, and printer I/O. This EPROM is bank switched in and out of position in low memory as needed by the BIOS.

The Holmes board lets you read and write to disks formatted by the IBM PC, Kaypro II, Xerox 820, Xerox 820-II, Osborne-I, Zenith Z-100, Freedom Technology (Model III CP/M system), Memory Merchant Shuffleboard III (Model III CP/M), Omikron (Model I CP/M), and Morrow Micro-Decision.

If your computer has 8-inch disk drive capability, Holmes CP/M lets you use the 8-inch drives without patching. Eight-inch drive support includes the ability to read and write to disks formatted in IBM 3740 single density, Xerox 820 single density, and Xerox 820-II double density.

The Holmes board is available in two versions. The first, and cheaper, version has a 16K bank of RAM and uses the Model III's 48K RAM for the rest of the RAM it needs. The other version comes with 64K of on-board RAM and doesn't use the Model III RAM except as a RAM disk accessible from CP/M for extremely fast data storage and retrieval.

When I wrote this review, no known incompatibilities existed between Holmes CP/M and other CP/M software. You can even run UCSD Pascal with the Holmes board.

Shuffleboard III

The Shuffleboard III (see Photo 3), a 64K CP/M 2.2 system, is as easy as the others to install. The only difference is that you remove one of the Model III memory chips, U25, and replace it with a 16-pin plug attached to the Shuffleboard unit. After installing the board, you put your TRSDOS disk or your CP/M disk in drive zero and press reset. Level II Basic is still available, of course.

Like Omikron, Memory Merchant has provided MBasic for Basic programmers. You can also purchase an 80 by 24 video board separately for $275.

Actually, you have several hardware options with the Shuffleboard: If you want to, you can move a jumper on the board from its "Automatic boot from drive zero" position to another. This clears the screen and asks you which drive contains the system disk (press zero for drive zero, one for drive 1, and so on).

Another board option lets you select operation of either a 2716 ROM, or 2732 or 2764 ROMs. The manual doesn't discuss this option, so it's probably being reserved for future developments and enhancements.

A unique feature of Shuffleboard is the ability to switch between CP/M and TRSDOS using the appropriate command. If you're in TRSDOS and want to go to CP/M, enter the DOS command CPM. From the CP/M prompt, enter DOS. These are memory erasing changes: Anything in memory is zeroed out, so you can't transfer between the two operating systems and maintain in-memory data integrity.

In its present form, Shuffleboard CP/M is compatible with 35-, 40-, 77-, and 80-track 5¼-inch disk drives. With the addition of the Memory Merchant disk controller, you can use 8-inch disk drives.

As with the Omikron board, you can use your Model III as a slave to a remote terminal, using the remote terminal for input instead of the keyboard.

See List of Advertisers on Page 291

And unlike the other two CP/M systems, Shuffleboard includes the Auto command as a valid command, which operates just like the TRSDOS Auto command. The length of the command is limited to one screen line (64 characters).

Memory Merchant CP/M is configured to expect the CP/M drive A to be the physical drive zero, drive B to be drive 1, drive C to be drive 2, drive D to be drive 3, and drive E to be nonexistent. Drive E invokes a special Memory Merchant CP/M utility called a virtual drive. You also have drives F–I available as single-density drives.

The first two logical drives (A and B) are 5¼-inch, 40-track, double-density drives. The third drive is configured as an 8-inch, 77-track, double-density drive. You can reorganize the CP/M drive map to match any setup you have, including using physical drive 4 as the boot-up drive, logical drive A. You do all this by using a disk drive map that lets you match any of the logical drive parameters to the physical drives.

Now for the virtual drive. Memory Merchant has altered CP/M so you can logically change a disk drive assignment while running CP/M, a procedure not allowed with the other CP/M systems. This virtual drive capability means that you can make single-drive copies.

If you have two 5¼-inch drives and one 8-inch drive, you can use the virtual drive assignment to copy a file from one 8-inch disk to another. Otherwise, you'd have to copy the file to a 5¼-inch disk, change disks in the 8-inch drive, invoke a warm-boot on that drive, then copy the file from the 5¼-inch drive to the new 8-inch disk.

This virtual assignment capability is reflected in the use of logical drives F–I. Physical drives 1–3 are given two logical drive numbers, B–D and F–H. Drive 1 is both logical drive B and logical drive F. The difference is that drive B is a double-density drive and drive F is a single-density drive.

You can copy files from or to a single-density disk without going to the disk drive assignment table and changing one of the drives to be logically single-density. To let you copy files between single density and double density on the same drive, logical drive I is used, just as logical drive E is used for single-drive, double-density file copying.

"All three boards are easy to install."

Speaking of copying, the Shuffleboard can read and write to Osborne I, Xerox 820, and IBM PC formatted disks.

Three manuals come with the Shuffleboard III. The user's manual is a 77-page, 8- by 11-inch printed booklet that covers all phases of board installation and CP/M use. The second manual is a bound edition of the new Digital Research documentation on CP/M.

The last manual is the Microsoft MBasic instruction book. Unlike the previous DRI manuals, these books are well written and easy to understand. Both books top 250 pages in length.

Summary

All three boards are easy to install with only screwdrivers and a nail file. If you follow the instructions carefully, you should have no problems with the installation.

One disadvantage shared by all three boards is that they make it difficult for the CPU RFI shield to fit properly. The clearance between the RFI shield and the CPU board is almost, but not quite, enough room for the new piggy-back boards. On my Model III, I could only get the screws on the left side (looking at the CPU board from the rear of the computer) to go into place.

Putting the top and right side screws in place put a good deal of stress on the two boards, and actually forced the RFI shield to bulge. However, the three screws I used easily keep the shield in place, and should maintain the cage design of the Model III.

At present, the Holmes board has the largest base of CP/M disk formats that it can read and write, but the others are rapidly catching up. And all three companies offer excellent customer support, with customer hotline phone numbers.

As you can see, the final decision on which board is best depends on your needs. As a purely personal response, the Holmes Engineering board is my favorite; it seems to offer the most for my needs. ■

Contact Terry Kepner c/o 80 Micro, *80 Pine St., Peterborough, NH 03458.*

every month.

ALL THE NEWS YOU NEED FOR THE 80'S

80 Micro is news. News like our in-depth coverage of Tandy's Model 100 debut in Boston and its impact on the market. News like U.S. Senator Paul Tsongas' speech on Asian competition and the importance of innovation. News like what's happening at Texas Instruments and why. You'll agree that **80's "NEWS THIS MONTH"** column is the news you need to stay informed.

80 is also a practical journal. It gives you the information you need to expand your TRS-80 computing potential. Information like:

- New product reviews—save valuable time getting the facts and figures on the latest equipment releases.
- Hardware modifications—upgrade your computer and become more familiar with its functions while you save money and increase its value.
- Debugging techniques—**80 Micro** saves time with expert solutions to common and uncommon problems.
- User-application programs—written by readers like you who need programs to maximize the productivity of their machines.

And **80 Micro** doesn't stop there. Every month, from cover to cover, you'll get top rate REVIEWS: whether you're buying or just looking, **80** keeps you informed on what state-of-the-art can do for you.

If you want to get ahead with your TRS-80, get **80 Micro**, the news source of the 80's.

YES! Give me the news I need to put me ahead and help me stay there. Send me 12 issues of 80 Micro for $24.97.

☐ MC ☐ VISA ☐ AE ☐ CHECK/MO ☐ BILL ME

Card # ______________________ Exp. Date ________

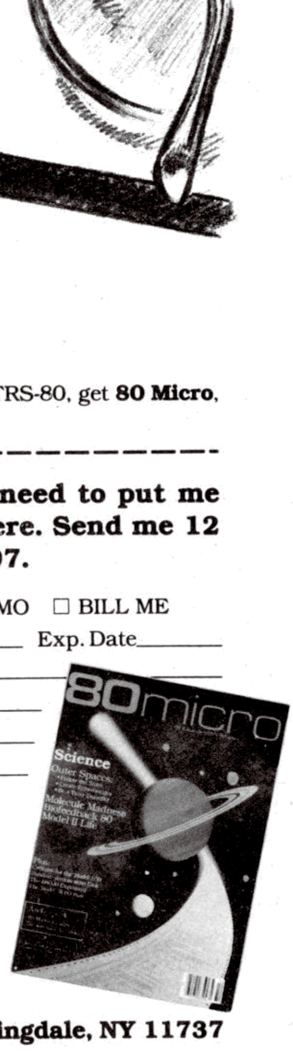
Signature ______________________________

Name ______________________

Address ______________________

City ______________ State ______________

Zip ______________



Canada and Mexico $27.97, 1 year only, US funds drawn on US bank.
Foreign surface $44.97, 1 year only, US funds drawn on US bank.

Please allow 6-8 weeks for delivery.

*TRS-80 is a trademark of Radio Shack, a division of Tandy Corp.

33DP8

80 Micro • PO Box 981 • Farmingdale, NY 11737

Protected Tape Programs

by Dan Robinson

This pair of short routines safeguards your cassette programs against predators.

ProteK (with a K; see Program Listing 1) makes your system tape auto-start so spies can't jump into a monitor program. It adds a leader to your tape which befuddles copy programs like the RSM series or LMOFFSET by sending a false checksum. A double delay loop foils other copy cats that swallow programs until they think they've devoured the last byte.

The program includes pitfalls for those who load a monitor first and count on getting to your programs following a reboot: The entry points for TBUG, DEBUG, and RSM2 16K, 32K, and 48K will take them back to the MEMORY SIZE? prompt.

Protect your tape programs from predators with these two routines that prohibit copying.

ProteC (with a C; see Program Listing 2) converts your Basic program into a system tape with an auto-start feature and the same booby traps as its companion program. Furthermore, the program turns the trace off (TROFF) and disables the break key so that no one can list your program to the screen or a printer.

Instructions

To safeguard a Basic program, you can load ProteC either before or after your Basic file. As it completes its load, ProteC displays detailed instructions on the screen. The program then returns to the Basic READY prompt, and you're set to go.

ProteK does not list the instructions to the screen; you load it in series after the system program you wish to encode. When you load ProteK, it automatically boots up, asking for the name of the new system tape and the hexadecimal start/end/transfer address. The program prompts you to prepare the tape recorder, and then writes your encoded program.

Both ProteC and ProteK change some pointers and addresses, so once you've encoded your program it's wise to reset your TRS-80. ■

Contact Dan Robinson at 1625 Higgins Way, Pacifica, CA 94044.

The Key Box

Model I
16K RAM
Assembly Language
Editor/Assembler
Tape Recorder

Program Listing 1. ProteK routine.

```
                   ;PROTEK 2.0---CASSETTE---SYSTEM TAPES ONLY
41E2                       ORG     41E2H   ;SYSTEM RELAY FOR AUTO START
41E2 E9                    DEFB    0E9H
7E00                       ORG     7E00H   ;PROTEK
7E00               START   EQU     $
7E00 CDC901                CALL    01C9H   ;CLEAR SCREEN
7E03 21797F                LD      HL,MSG1 ;DISPLAY PGM NAME PROMPT
7E06 CDA728                CALL    28A7H
7E09 3E0D                  LD      A,0DH
7E0B CD3A03                CALL    033AH
7E0E 2AA740                LD      HL,(40A7H)
7E11 0615                  LD      B,21
7E13 CDD905                CALL    05D9H   ;INPUT NAME & ADDRESSES
7E16 116D7F                LD      DE,NAME
7E19 7E            NAME1   LD      A,(HL)
7E1A FE2F                  CP      2FH     ; "/"
7E1C 2805                  JR      Z,NAME2
7E1E 12                    LD      (DE),A
7E1F 23                    INC     HL
7E20 13                    INC     DE
7E21 18F6                  JR      NAME1
7E23 CD1D7F        NAME2   CALL    HEX
7E26 32747F                LD      (STADR+1),A
7E29 CD1D7F                CALL    HEX
7E2C 32737F                LD      (STADR),A
7E2F 23                    INC     HL
7E30 CD1D7F                CALL    HEX
7E33 32767F                LD      (ENDADR+1),A
7E36 CD1D7F                CALL    HEX
7E39 32757F                LD      (ENDADR),A
7E3C 23                    INC     HL
7E3D CD1D7F                CALL    HEX
7E40 32787F                LD      (TRFADR+1),A
7E43 CD1D7F                CALL    HEX
7E46 32777F                LD      (TRFADR),A
7E49 3E0D                  LD      A,0DH
7E4B CD3A03                CALL    033AH
7E4E 21AA7F                LD      HL,MSG2 ;DISPLAY PREPARE RECORDER
7E51 CDA728                CALL    28A7H
7E54 010000                LD      BC,0000
7E57 CD6000                CALL    0060H
7E5A B7                    OR      A
7E5B 3A4038        READY   LD      A,(3840H)       ;CHECK KEYBOARD FOR <ENTER>
7E5E FE01                  CP      01H
7E60 2807                  JR      Z,REC
7E62 FE04                  CP      04      ;CHECK KEYBOARD FOR <BREAK>
7E64 CA007E                JP      Z,START ;RETURN TO BASIC
```

Listing 1 continued

Listing 1 continued

```
7E67 18F2     00570        JR      READY
7E69 F3       00580 REC    DI
7E6A 3E2A     00590        LD      A,2AH
7E6C 323E3C   00600        LD      (3C3EH),A       ;STAR
7E6F 3E00     00610        LD      A,0
7E71 CD1202   00620        CALL    0212H   ;INITIALIZE RECORDER #1
7E74 CD8702   00630        CALL    0287H   ;WRITE LEADER & SYNC BYTE
7E77 3E55     00640        LD      A,55H   ;SYSTEM TAPE ID
7E79 CD6402   00650        CALL    0264H   ;WRITE BYTE
7E7C 116D7F   00660        LD      DE,NAME
7E7F 0606     00670        LD      B,6
7E81 1A       00680 WRTNAM LD      A,(DE)
7E82 CD6402   00690        CALL    0264H
7E85 13       00700        INC     DE
7E86 10F9     00710        DJNZ    WRTNAM  ;WRITE NAME
7E88 3E3C     00720        LD      A,3CH   ;DISABLE LMOFFSET
7E8A CD6402   00730        CALL    0264H
7E8D 3E01     00740        LD      A,1
7E8F CD6402   00750        CALL    0264H
7E92 3E00     00760        LD      A,00H
7E94 CD6402   00770        CALL    0264H
7E97 3E00     00780        LD      A,00H
7E99 CD6402   00790        CALL    0264H
7E9C 3E00     00800        LD      A,0
7E9E CD6402   00810        CALL    0264H
7EA1 CD6402   00820        CALL    0264H   ;OUTPUT FALSE CHECKSUM
7EA4 213E3C   00830        LD      HL,3C3EH
7EA7 DD21E07F 00840        LD      IX,STAR
7EAB CD417F   00850        CALL    SET
7EAE 010000   00860        LD      BC,0000 ;DEFEAT TIMERS
7EB1 CD6000   00870        CALL    0060H
7EB4 CD6000   00880        CALL    0060H
7EB7 21A043   00890        LD      HL,43A0H        ;TBUG
7EBA CD3D7F   00900        CALL    BLOCK
7EBD 21807F   00910        LD      HL,7F80H        ;RSM2 & 2D 16K
7EC0 CD3D7F   00920        CALL    BLOCK
7EC3 2180BF   00930        LD      HL,0BF80H       ;RSM2D 32K
7EC6 CD3D7F   00940        CALL    BLOCK
7EC9 2180FF   00950        LD      HL,0FF80H       ;RSM2D 48K
7ECC CD3D7F   00960        CALL    BLOCK
7ECF 210F40   00970        LD      HL,400FH        ;DEBUG
7ED2 CD3D7F   00980        CALL    BLOCK
              00990 ;ADD BLOCKING ROUTINES HERE--LD HL,ADDR CALL BLOCK
7ED5 2A757F   01000        LD      HL,(ENDADR)
7ED8 ED5B737F 01010        LD      DE,(STADR)
7EDC B7       01020        OR      A
7EDD ED52     01030        SBC     HL,DE
7EDF EB       01040        EX      DE,HL
7EE0 E5       01050        PUSH    HL
7EE1 DDE1     01060        POP     IX
7EE3 7A       01070        LD      A,D
7EE4 FE00     01080        CP      0
7EE6 2808     01090        JR      Z,WRT2
7EE8 0600     01100 WRT1   LD      B,0
7EEA CD437F   01110        CALL    PUNCH
7EED 15       01120        DEC     D
7EEE 20F8     01130        JR      NZ,WRT1
7EF0 7B       01140 WRT2   LD      A,E
7EF1 FE00     01150        CP      0
7EF3 CAFA7E   01160        JP      Z,WRT3
7EF6 43       01170        LD      B,E
7EF7 CD437F   01180        CALL    PUNCH
7EFA 0601     01190 WRT3   LD      B,1
7EFC DD216C7F 01200        LD      IX,LIST
7F00 21E241   01210        LD      HL,41E2H        ;SYSTEM/AUTO-START RELAY
7F03 CD437F   01220        CALL    PUNCH
7F06 ED5B777F 01230        LD      DE,(TRFADR)
7F0A 3E78     01240        LD      A,78H   ;TRANSFER BLOCK
7F0C CD6402   01250        CALL    0264H
7F0F 7B       01260        LD      A,E
7F10 CD6402   01270        CALL    0264H
7F13 7A       01280        LD      A,D
7F14 CD6402   01290        CALL    0264H
7F17 CDF801   01300        CALL    01F8H   ;TURN OFF RECORDER
7F1A C3007E   01310        JP      START
7F1D 23       01320 HEX    INC     HL      ;ASCII TO HEX
7F1E 7E       01330        LD      A,(HL)
7F1F D630     01340        SUB     30H
7F21 FE0A     01350        CP      10
7F23 F43A7F   01360        CALL    P,ADJ
7F26 CB27     01370        SLA     A
7F28 CB27     01380        SLA     A
7F2A CB27     01390        SLA     A
7F2C CB27     01400        SLA     A
7F2E 47       01410        LD      B,A
7F2F 23       01420        INC     HL
7F30 7E       01430        LD      A,(HL)
7F31 D630     01440        SUB     30H
7F33 FE0A     01450        CP      10
7F35 F43A7F   01460        CALL    P,ADJ
7F38 80       01470        ADD     A,B
7F39 C9       01480        RET
7F3A D607     01490 ADJ    SUB     7
7F3C C9       01500        RET
7F3D DD216B7F 01510 BLOCK  LD      IX,RESET
7F41 0601     01520 SET    LD      B,1
7F43 3E3C     01530 PUNCH  LD      A,3CH   ;BLOCK SYNC
7F45 CD6402   01540        CALL    0264H   ;WRITE BYTE
7F48 78       01550        LD      A,B     ;BYTE COUNT
7F49 CD6402   01560        CALL    0264H
7F4C 7D       01570        LD      A,L     ;LSB ADDRESS
7F4D CD6402   01580        CALL    0264H
7F50 4F       01590        LD      C,A     ;SAVE FOR CHECKSUM
7F51 7C       01600        LD      A,H     ;MSB ADDRESS
7F52 CD6402   01610        CALL    0264H
7F55 81       01620        ADD     A,C
7F56 4F       01630        LD      C,A
```

Listing 1 continued

Listing 1 continued

```
7F57 DD7E00     01640 PUN1     LD      A,(IX)
7F5A CD6402     01650          CALL    0264H
7F5D 23         01660          INC     HL
7F5E DD23       01670          INC     IX
7F60 81         01680          ADD     A,C
7F61 4F         01690          LD      C,A
7F62 10F3       01700          DJNZ    PUN1
7F64 CD6402     01710          CALL    0264H    ;OUTPUT CHECKSUM
7F67 CD2C02     01720          CALL    022CH    ;BLINK STAR
7F6A C9         01730          RET
7F6B 76         01740 RESET    DEFB    76H      ;HALT CODE
7F6C E9         01750 LIST     DEFB    0E9H
7F6D 20         01760 NAME     DEFB    20H
7F6E 20         01770          DEFB    20H
7F6F 20         01780          DEFB    20H
7F70 20         01790          DEFB    20H
7F71 20         01800          DEFB    20H
7F72 20         01810          DEFB    20H
0002            01820 STADR    DEFS    2        ;STORE START ADDRESS
0002            01830 ENDADR   DEFS    2        ;STORE END ADDRESS
0002            01840 TRFADR   DEFS    2        ;STORE ENTRY ADDRESS
7F79 45         01850 MSG1     DEFM    'ENTER PROGRAM NAME/START/END/ENTRY HEX ADDR
ESSES'
      4E 54 45 52 20 50 52 4F
      47 52 41 4D 20 4E 41 4D
      45 2F 53 54 41 52 54 2F
      45 4E 44 2F 45 4E 54 52
      59 20 48 45 58 20 41 44
      44 52 45 53 53 45 53
7FA9 00         01860          DEFB    0
7FAA 20         01870 MSG2     DEFM    '      PREPARE RECORDER AND PRESS RECORD & PL
AY <ENTER>'
      20 20 20 20 50 52 45 50
      41 52 45 20 52 45 43 4F
      52 44 45 52 20 41 4E 44
      20 50 52 45 53 53 20 52
      45 43 4F 52 44 20 26 20
      50 4C 41 59 20 3C 45 4E
      54 45 52 3E
7FDF 00         01880          DEFB    0
7FE0 2A         01890 STAR     DEFB    2AH
7E00            01900          END     START
00000 TOTAL ERRORS
```

Program Listing 2. ProteC routine.

```
                00100 ;PROTEC 1.1---BASIC CASSETTE TAPES ONLY
41E2            00110          ORG     41E2H    ;SYSTEM RELAY FOR AUTO START
41E2 E9         00120          DEFB    0E9H
40B1            00130          ORG     40B1H    ;BASIC TOP OF MEMORY PTR
40B1 1F7E       00140          DEFW    START-1
418E            00150          ORG     418EH    ;NAME VECTOR
418E C3         00160          DEFB    0C3H     ;JUMP
418F 207E       00170          DEFW    START
4049            00180          ORG     4049H    ;MEMORY SIZE
4049 1F7E       00190          DEFW    START-1
7E20            00200          ORG     7E20H    ;PROTEK
7E20            00210 START    EQU     $
7E20 CDC901     00220          CALL    01C9H    ;CLEAR SCREEN
7E23 31287E     00230          LD      SP,START+8
7E26 21837F     00350          LD      HL,MSG1 ;DISPLAY PGM NAME PROMPT
7E29 CDA728     00360          CALL    28A7H
7E2C 3E0D       00370          LD      A,0DH
7E2E CD3A03     00380          CALL    033AH
7E31 2AA740     00390          LD      HL,(40A7H)
7E34 0606       00400          LD      B,6
7E36 CDD905     00410          CALL    05D9H    ;INPUT NAME
7E39 117D7F     00420          LD      DE,NAME
7E3C 7E         00440 NAME1    LD      A,(HL)
7E3D 12         00450          LD      (DE),A
7E3E 23         00460          INC     HL
7E3F 13         00470          INC     DE
7E40 10FA       00480          DJNZ    NAME1
7E42 3E0D       00490          LD      A,0DH
7E44 CD3A03     00500          CALL    033AH
7E47 21967F     00510          LD      HL,MSG2 ;DISPLAY PREPARE RECORDER
7E4A CDA728     00520          CALL    28A7H
7E4D 010000     00522          LD      BC,0000
7E50 CD6000     00524          CALL    0060H
7E53 B7         00530          OR      A
7E54 3A4038     00540 READY    LD      A,(3840H)        ;CHECK KEYBOARD FOR <ENTER>
7E57 FE01       00550          CP      01H
7E59 2807       00560          JR      Z,REC
7E5B FE04       00570          CP      04       ;CHECK KEYBOARD FOR <BREAK>
7E5D CA191A     00580          JP      Z,1A19H ;RETURN TO BASIC
7E60 18F2       00590          JR      READY
7E62 F3         00600 REC      DI
7E63 3E2A       00610          LD      A,2AH
7E65 323E3C     00620          LD      (3C3EH),A        ;STAR
7E68 3E00       00630          LD      A,0
7E6A CD1202     00650          CALL    0212H    ;INITIALIZE RECORDER #1
7E6D CD8702     00660          CALL    0287H    ;WRITE LEADER & SYNC BYTE
7E70 3E55       00670          LD      A,55H    ;SYSTEM TAPE ID
7E72 CD6402     00680          CALL    0264H    ;WRITE BYTE
7E75 117D7F     00690          LD      DE,NAME
7E78 0606       00700          LD      B,6
7E7A 1A         00710 WRTNAM   LD      A,(DE)
7E7B CD6402     00720          CALL    0264H
7E7E 13         00730          INC     DE
7E7F 10F9       00740          DJNZ    WRTNAM   ;WRITE NAME
7E81 3E3C       00800          LD      A,3CH    ;DISABLE TRACE
```

Listing 2 continued

Listing 2 continued

```
7E83 CD6402    00810          CALL    0264H
7E86 3E01      00820          LD      A,1
7E88 CD6402    00830          CALL    0264H
7E8B 3E1B      00840          LD      A,1BH
7E8D CD6402    00850          CALL    0264H
7E90 3E41      00860          LD      A,41H
7E92 CD6402    00870          CALL    0264H
7E95 3E00      00880          LD      A,0
7E97 CD6402    00890          CALL    0264H
7E9A CD6402    00900          CALL    0264H   ;OUTPUT FALSE CHECKSUM
7E9D 213E3C    00910          LD      HL,3C3EH
7EA0 DD21CC7F  00930          LD      IX,STAR
7EA4 CD477F    00940          CALL    SET
7EA7 010000    00942          LD      BC,0000 ;DEFEAT TIMERS
7EAA CD6000    00944          CALL    0060H
7EAD CD6000    00946          CALL    0060H
7EB0 210000    00950          LD      HL,0000 ;LMOFFSET
7EB3 CD437F    00970          CALL    BLOCK
7EB6 21A043    00980          LD      HL,43A0H        ;TBUG
7EB9 CD437F    00990          CALL    BLOCK
7EBC 21807F    01000          LD      HL,7F80H        ;RSM2 & 2D, 16K
7EBF CD437F    01010          CALL    BLOCK
7EC2 2180BF    01020          LD      HL,0BF80H       ;RSM2D 32K
7EC5 CD437F    01030          CALL    BLOCK
7EC8 2180FF    01040          LD      HL,0FF80H       ;RSM2D 48K
7ECB CD437F    01050          CALL    BLOCK
7ECE 210F40    01052          LD      HL,400FH        ;DEBUG
7ED1 CD437F    01054          CALL    BLOCK
7ED4 210C40    01056          LD      HL,400CH        ;DISABLE BREAK KEY
7ED7 DD217A7F  01058          LD      IX,BREAK
7EDB 0603      01060          LD      B,3
7EDD CD497F    01062          CALL    PUNCH
7EE0 2AF940    01064          LD      HL,(40F9H)      ;SIMPLE VARIABLES PTR
7EE3 ED5BA440  01066          LD      DE,(40A4H)      ;START OF BASIC PGM PTR
7EE7 B7        01068          OR      A
7EE8 ED52      01070          SBC     HL,DE
7EEA 2B        01072          DEC     HL
7EEB EB        01074          EX      DE,HL
7EEC E5        01076          PUSH    HL
7EED DDE1      01078          POP     IX
7EEF 7A        01080          LD      A,D
7EF0 FE00      01082          CP      0
7EF2 2808      01084          JR      Z,WRT2
7EF4 0600      01100 WRT1     LD      B,0
7EF6 CD497F    01110          CALL    PUNCH
7EF9 15        01120          DEC     D
7EFA 20F8      01130          JR      NZ,WRT1
7EFC 7B        01140 WRT2     LD      A,E
7EFD FE00      01150          CP      0
7EFF CA067F    01160          JP      Z,WRT3
7F02 43        01170          LD      B,E
7F03 CD497F    01180          CALL    PUNCH
7F06 E5        01190 WRT3     PUSH    HL      ;SAVE FOR TRANSFER
7F07 DD21727F  01200          LD      IX,LIST
7F0B 0607      01210          LD      B,7
7F0D CD497F    01220          CALL    PUNCH
7F10 21E241    01240          LD      HL,41E2H        ;SYSTEM/AUTO-START RELAY
7F13 CD477F    01250          CALL    SET
7F16 0602      01290          LD      B,2
7F18 21F940    01300          LD      HL,40F9H        ;SIMPLE VARIABLES PTR
7F1B E5        01310          PUSH    HL
7F1C DDE1      01312          POP     IX
7F1E CD497F    01320          CALL    PUNCH
7F21 21A440    01330          LD      HL,40A4H        ;START OF BASIC PTR
7F24 E5        01340          PUSH    HL
7F25 DDE1      01342          POP     IX
7F27 0602      01350          LD      B,2
7F29 CD497F    01360          CALL    PUNCH
7F2C D1        01370          POP     DE      ;RECOVER RUN BLOCK
7F2D 3E78      01380          LD      A,78H   ;TRANSFER BLOCK
7F2F CD6402    01390          CALL    0264H
7F32 7B        01400          LD      A,E     ;RUN ROUTINE
7F33 CD6402    01410          CALL    0264H
7F36 7A        01420          LD      A,D
7F37 CD6402    01430          CALL    0264H
7F3A CDF801    01440          CALL    01F8H   ;TURN OFF RECORDER
7F3D CDC901    01480          CALL    01C9H   ;CLEAR SCREEN
7F40 C3191A    01490          JP      1A19H   ;RETURN TO BASIC
7F43 DD21717F  01500 BLOCK    LD      IX,RESET
7F47 0601      01510 SET      LD      B,1
7F49 3E3C      01520 PUNCH    LD      A,3CH   ;BLOCK SYNC
7F4B CD6402    01530          CALL    0264H   ;WRITE BYTE
7F4E 78        01540          LD      A,B     ;BYTE COUNT
7F4F CD6402    01550          CALL    0264H
7F52 7D        01560          LD      A,L     ;LSB ADDRESS
7F53 CD6402    01570          CALL    0264H
7F56 4F        01580          LD      C,A     ;SAVE FOR CHECKSUM
7F57 7C        01590          LD      A,H     ;MSB ADDRESS
7F58 CD6402    01600          CALL    0264H
7F5B 81        01610          ADD     A,C
7F5C 4F        01620          LD      C,A
7F5D DD7E00    01630 PUN1     LD      A,(IX)
7F60 CD6402    01640          CALL    0264H
7F63 23        01650          INC     HL
7F64 DD23      01660          INC     IX
7F66 81        01670          ADD     A,C
7F67 4F        01680          LD      C,A
7F68 10F3      01690          DJNZ    PUN1
7F6A CD6402    01700          CALL    0264H   ;OUTPUT CHECKSUM
7F6D CD2C02    01710          CALL    022CH   ;BLINK STAR
7F70 C9        01720          RET
7F71 76        01722 RESET    DEFB    76H     ;HALT CODE
7F72 21        01730 LIST     DEFB    21H
7F73 1E        01740          DEFB    1EH
7F74 1D        01750          DEFB    1DH
7F75 E5        01760          DEFB    0E5H
7F76 C3        01770          DEFB    0C3H
```

Listing 2 continued

Listing 2 continued

```
7F77 5D          01780          DEFB     5DH
7F78 1B          01790          DEFB     1BH
7F79 E9          01800          DEFB     0E9H
7F7A C3          01810 BREAK    DEFB     0C3H
7F7B 98          01820          DEFB     98H
7F7C 0A          01830          DEFB     0AH
7F7D 20          01840 NAME     DEFB     20H
7F7E 20          01842          DEFB     20H
7F7F 20          01844          DEFB     20H
7F80 20          01846          DEFB     20H
7F81 20          01848          DEFB     20H
7F82 20          01849          DEFB     20H
7F83 45          01850 MSG1     DEFM     'ENTER PROGRAM NAME'
     4E 54 45 52 20 50 52 4F
     47 52 41 4D 20 4E 41 4D
     45
7F95 00          01860          DEFB     0
7F96 20          01870 MSG2     DEFM     '     PREPARE RECORDER AND PRESS RECORD & PL
AY <ENTER>'
     20 20 20 20 50 52 45 50
     41 52 45 20 52 45 43 4F
     52 44 45 52 20 41 4E 44
     20 50 52 45 53 53 20 52
     45 43 4F 52 44 20 26 20
     50 4C 41 59 20 3C 45 4E
     54 45 52 3E
7FCB 00          01880          DEFB     0
7FCC 2A          01890 STAR     DEFB     2AH
3C00             01910          ORG      3C00H   ;WRITE INSTRUCTIONS
3C00 20          01920          DEFM     '                              PROTEC
                     '
     20 20 20 20 20 20 20 20
     20 20 20 20 20 20 20 20
     20 20 20 20 20 20 20 20
     20 20 20 20 50 52 4F 54
     45 43 20 20 20 20 20 20
     20 20 20 20 20 20 20 20
     20 20 20 20 20 20 20 20
     20 20 20 20 20 20 20
3C3F 20          01930          DEFM     '     WHEN YOU ARE READY TO ENCODE YOUR BASI
C PROGRAM, TYPE NAME'
     20 20 20 20 57 48 45 4E
     20 59 4F 55 20 41 52 45
     20 52 45 41 44 59 20 54
     4F 20 45 4E 43 4F 44 45
     20 59 4F 55 52 20 42 41
     53 49 43 20 50 52 4F 47
     52 41 4D 2C 20 54 59 50
     45 20 4E 41 4D 45
3C7E 20          01940          DEFM     ' <ENTER>.  YOU WILL BE PROMPTED TO ENTER U
P TO SIX CHARACTERS '
     20 3C 45 4E 54 45 52 3E
     2E 20 20 59 4F 55 20 57
     49 4C 4C 20 42 45 20 50
     52 4F 4D 50 54 45 44 20
     54 4F 20 45 4E 54 45 52
     20 55 50 20 54 4F 20 53
     49 58 20 43 48 41 52 41
     43 54 45 52 53 20
3CBD 20          01950          DEFM     '   UNDER WHICH YOUR ENCODED PROGRAM WILL OP
ERATE AS A SYSTEM TA'
     20 20 55 4E 44 45 52 20
     57 48 49 43 48 20 59 4F
     55 52 20 45 4E 43 4F 44
     45 44 20 50 52 4F 47 52
     41 4D 20 57 49 4C 4C 20
     4F 50 45 52 41 54 45 20
     41 53 20 41 20 53 59 53
     54 45 4D 20 54 41
3CFC 50          01960          DEFM     'PE  <ENTER>.  YOU WILL THEN BE PROMPTED TO
PREPARE THE RECORDER'
     45 20 20 3C 45 4E 54 45
     52 3E 2E 20 20 59 4F 55
     20 57 49 4C 4C 20 54 48
     45 4E 20 42 45 20 50 52
     4F 4D 50 54 45 44 20 54
     4F 20 50 52 45 50 41 52
     45 20 54 48 45 20 52 45
     43 4F 52 44 45 52
3D3B 2C          01970          DEFM     ',    PRESSING RECORD AND PLAY <ENTER>. YOUR
 BASIC PROGRAM WILL'
     20 20 20 20 50 52 45 53
     53 49 4E 47 20 52 45 43
     4F 52 44 20 41 4E 44 20
     50 4C 41 59 20 3C 45 4E
     54 45 52 3E 2E 20 59 4F
     55 52 20 42 41 53 49 43
     20 50 52 4F 47 52 41 4D
     20 57 49 4C 4C
3D79 20          01980          DEFM     ' BE    ENCODED AS A SYSTEM TAPE. TO RUN YOU
R TAPE, TYPE SYSTEM'
     42 45 20 20 20 20 45 4E
     43 4F 44 45 44 20 41 53
     20 41 20 53 59 53 54 45
     4D 20 54 41 50 45 2E 20
     54 4F 20 52 55 4E 20 59
     4F 55 52 20 54 41 50 45
     2C 20 54 59 50 45 20 53
     59 53 54 45 4D
3DB7 20          01990          DEFM     ' <ENTER>.    AT THE PROMPT *? ENTER THE NAM
E UNDER WHICH THE'
     3C 45 4E 54 45 52 3E 2E
     20 20 20 20 41 54 20 54
     48 45 20 50 52 4F 4D 50
     54 20 2A 3F 20 45 4E 54
     45 52 20 54 48 45 20 4E
```

Listing 2 continued

Listing 2 continued

```
      41 4D 45 20 55 4E 44 45
      52 20 57 48 49 43 48 20
      54 48 45
3DF3 20          02000          DEFM     ' PROGRAM WASENCODED <ENTER>.  YOUR ENCODED
 PROGRAM WILL LOAD'
      20 50 52 4F 47 52 41 4D
      20 57 41 53 45 4E 43 4F
      44 45 44 20 3C 45 4E 54
      45 52 3E 2E 20 20 59 4F
      55 52 20 45 4E 43 4F 44
      45 44 20 50 52 4F 47 52
      41 4D 20 57 49 4C 4C 20
      4C 4F 41 44
3E30 20          02010          DEFM     ' AND AUTOMATIC- ALLY BEGIN TO OPERATE.  USE
RS WILL BE UNABLE TO'
      41 4E 44 20 41 55 54 4F
      4D 41 54 49 43 2D 20 41
      4C 4C 59 20 42 45 47 49
      4E 20 54 4F 20 4F 50 45
      52 41 54 45 2E 20 20 55
      53 45 52 53 20 57 49 4C
      4C 20 42 45 20 55 4E 41
      42 4C 45 20 54 4F
3E6F 20          02020          DEFM     '  LIST, LLIST OR USE THE BREAK KEY.  POPULA
R MONITORS SUCH AS'
      20 4C 49 53 54 2C 20 4C
      4C 49 53 54 20 4F 52 20
      55 53 45 20 54 48 45 20
      42 52 45 41 4B 20 4B 45
      59 2E 20 20 50 4F 50 55
      4C 41 52 20 4D 4F 4E 49
      54 4F 52 53 20 53 55 43
      48 20 41 53
3EAC 20          02030          DEFM     ' THE RSM2 SERIES,   SYSTEM COPY AND TBUG WI
LL BE DISABLED, AS W'
      54 48 45 20 52 53 4D 32
      20 53 45 52 49 45 53 2C
      20 20 20 53 59 53 54 45
      4D 20 43 4F 50 59 20 41
      4E 44 20 54 42 55 47 20
      57 49 4C 4C 20 42 45 20
      44 49 53 41 42 4C 45 44
      2C 20 41 53 20 57
3EEB 49          02040          DEFM     'ILL THE TRACE        FUNCTION.  YOU MAY NOW
 CLOAD YOUR BASIC PR'
      4C 4C 20 54 48 45 20 54
      52 41 43 45 20 20 20 20
      20 20 20 20 46 55 4E 43
      54 49 4F 4E 2E 20 20 59
      4F 55 20 4D 41 59 20 4E
      4F 57 20 43 4C 4F 41 44
      20 59 4F 55 52 20 42 41
      53 49 43 20 50 52
3F2A 4F          02050          DEFM     'OGRAM IF IT HAS NOT   ALREADY BEEN LOADED.
                '
      47 52 41 4D 20 49 46 20
      49 54 20 48 41 53 20 4E
      4F 54 20 20 20 41 4C 52
      45 41 44 59 20 42 45 45
      4E 20 4C 4F 41 44 45 44
      2E 20 20 20 20 20 20 20
      20 20 20 20 20 20 20 20
      20
4020             02060          ORG      4020H
4020 803F        02070          DEFW     3F80H
1A19             02080          END      1A19H    ;BASIC RE-ENTRY
00000 TOTAL ERRORS
```

BT ENTERPRISES

computer connection

SOFTWARE

for your TRS 80
MODEL I, III and 4

DOS PLUS 3.5 MODEL I/III

Better than standard Dos Plus. Added enhancements to make this the best system that will put you in total control.

Retails at $149.95 OUR PRICE *CALL FOR OUR LOW PRICE

DOS PLUS 4 MODEL 4

All the features of Dos Plus 3.5 and more. 80-column capability with full MicroSoft BASIC. 100% faster than TRSDOS 6.0.

Retails at $149.00 OUR PRICE *CALL FOR OUR LOW PRICE

CPM 2.2 FOR THE MODEL 4

From Montezuma Micro comes a fully implemented version of Digital Research's CPM 2.2. Now you can run all CPM-compatible programs on your Model 4 computer.

Retails at $199.00 OUR PRICE *CALL FOR OUR LOW PRICE

NEWSCRIPT

A complete word processing system based on the editing and text-formatting programs developed by IBM for use on mainframe time-sharing systems.

Retails at $110.00 OUR PRICE $85.00

SOS DISK TIMER

Speed-O-Scope drive speed test program used to check drive R.P.M. and to adjust TRS-80 Model III Disk Drives. Allows assurance and adjustment of your disk drives to proper operating speed.

Retails at $19.95 OUR PRICE $15.00

BASIC EDITOR

If you do a lot of editing of your BASIC programs, then this is the editor for you!!!

Retails at $29.95 OUR PRICE $24.95

***Manufacturers have asked us not to publish these low prices. Please call our toll free number for prices and also get on our free mailing list.**

MAS 80

The micro accounting system. Flexible, versatile, user-formatted business system designed for the novice and professional. AR/AP/GL/CK REG.

Retails at $599.00 OUR PRICE $499.00

M-ZAL

Release #3—more features than ever before. The most powerful editor assembler ever written. Full-screen editing, linking loader, full macro support and much more.

Retails at $99.95 OUR PRICE $75.00

ACCOUNTING PARTNER

Everything you need to make the system go—simply, efficiently, and conveniently. Easy to use, menu-driven programs, all performing integration calculations. Uses extensive error checking and leaves a complete audit trail. GL/AR/AP/PAYROLL Requires CP/M 2.2

Retails at $399.95 OUR PRICE $299.00

OFFER EXPIRES JANUARY 15th, 1984

Request your FREE Pascal 80 Trial Version with the purchase of any of these SPECIALS!

NEW CLASSICS SOFTWARE
239 FOX HILL ROAD DENVILLE NEW JERSEY 07834 TELEPHONE 201-625-8838

PASCAL 80
TRIAL VERSION

By Phelps Gates and Richard Koch
Program copyright 1981, 1983 by Phelps Gates.
This diskette will boot in a TRS-80 Model I or Model III

Features Supported in the Trial Version

Variable types: REAL, INTEGER, CHAR, BOOLEAN, arrays, sets, records and user defined types.
Constants: TRUE, FALSE, MAXINT, PI and user defined.
Operators: +, -, *, /, DIV, MOD, standard comparison and set operators, AND, OR and NOT.
Functions: ABS, SQR, SQRT, SIN, COS, ARCTAN, EXP, LN, ORD, PRED, SUCC, CHR, ODD, ROUND and TRUNC.
Procedures: READ, READLN, WRITE, WRITELN.
Structures: IF..THEN..ELSE, CASE, FOR..NEXT, WHILE..DO, REPEAT..UNTIL and GOTO.
Built in files: INPUT (keyboard), OUTPUT (screen) and LP (printer).

Available for $10 with no additional purchase.

BT ENTERPRISES
B.T. Enterprises Dept. 1-L
10 Carlough Rd.
Bohemia, N.Y. 11716

(516) 567-8155 (voice)
(516) 588-5836 (modem)

B T Enterprises is a division of Bi-Tech Enterprises Inc

Channels of Communication

by Dan Keen and Dave Dischert

Home satellite receiving dishes can bring you more than 140 cable stations free. This article explains dish technology and positioning.

You're sitting at home on a Friday night, looking for something entertaining to watch on television. But all that's on are reruns of The Dukes of Hazzard, a show on how to train your dog, and the Miss Teen U.S.A. pageant.

You do have an alternative—more than 140 TV and movie stations, each of which comes into your home free. All you need is a satellite receiver and a Model I or III.

In this article we discuss how satellite receivers work, how to get one, the costs and time involved, and how to use your Model I or III to position your receiving dish to pull in signals from different satellites.

Satellite Receiver Boom

Home satellite receivers are gaining in popularity. Currently, more than 100,000 U.S. homes have receivers. As more dishes are set up and newer, less expensive technology becomes available, dish receivers will drop in price and come within reach of almost everyone. As it is, satellite receiver systems are significantly less expensive today than they were five years ago. You can get a complete system now for as little as $2,500.

The Technology

Cable program signals travel around the United States via satellite (see Fig. 1). The transmitting station that generates the microwave signal beams it up to an orbiting satellite.

The satellite receives the signal, cleans it up, and rebroadcasts it to receiving stations scattered across the United States. These receiving stations then send the signal, via ground-based cables, to homes.

Home receiving dishes work just like their commercial counterparts. They, too, pick up signals from satellites, but instead of sending them out over commercial cables, they shunt the signal to your home television set.

Communications satellites orbit directly over the equator in a geostationary orbit at an altitude of 22,300 miles. Satellites occur in the North American continent in an equatorial arc of from roughly 83 to 135 degrees longitude, with a spread of about four degrees.

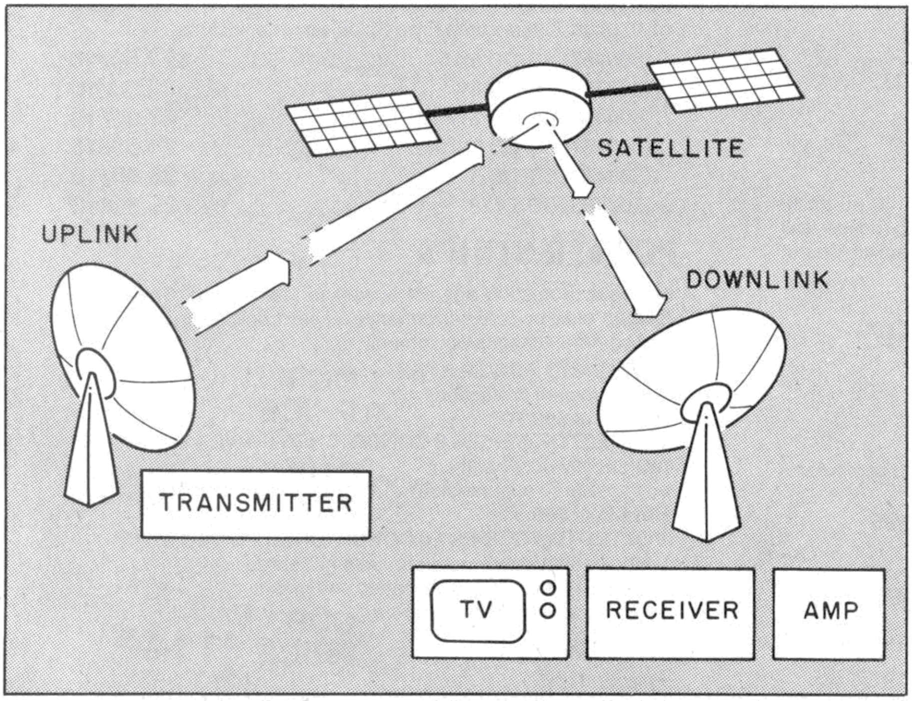


Figure 1. The basic components of a satellite TV system.

The Key Box

Model I and III
16K RAM Cassette Basic
32K RAM Disk Basic

Television Stations

Typically, satellites have either 12 or 24 channels, called transponders. Several communications satellites carry a full load (24 channels) devoted to television programming.

Some programs are free religious or commercial stations, such as Satellite News Channel and Music Television. Other channels may be illegal to receive, such as Home Box Office and Showtime, but comprehensive laws governing this medium are still being debated.

Satcom F3R, the most widely accessed satellite, is one of the few that dedicates all 24 transponders to television programming. It carries Nickelodeon, The Movie Channel, ESPN (entertainment sports), MTV (Music Television), Cable News Network, the Christian Broadcast Network, and superstations WGN and WTBS. The Program Listing contains all the stations available through Satcom F3R.

U.S. companies launch more satellites each year, providing a larger selection of programming services from which to choose.

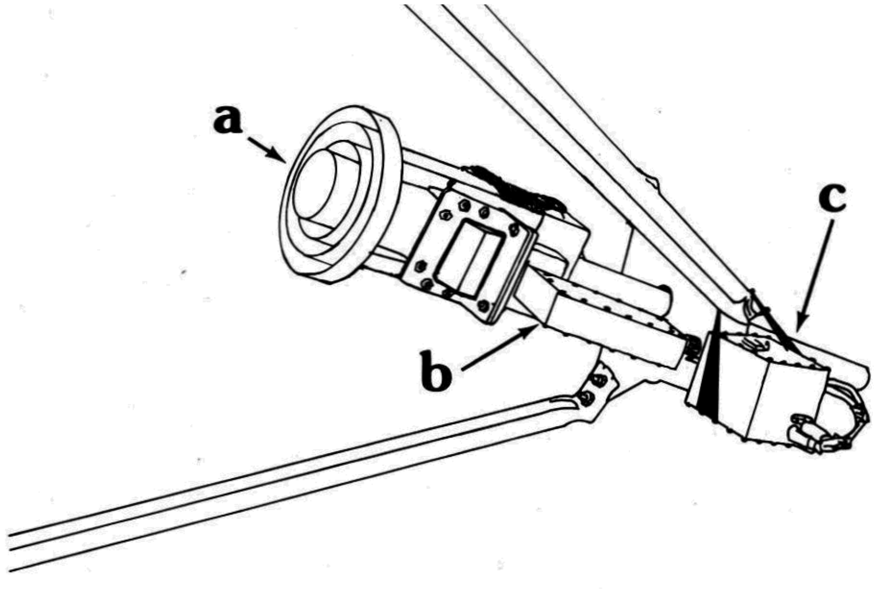


Figure 2. The feed assembly, consisting of the (a) antenna, (b) low-noise amplifier, and (c) downconverter.

Getting the Signal

Communications satellites send out polarized microwave beams in either a horizontal or vertical format. For example, RCA's Satcom F3R satellite has horizontal polarization on even transponders and vertical polarization on odd transponders. The Westar 3 satellite polarizes all its signals horizontally.

Even though all communications satellites (about 12 currently in operation) broadcast on the same frequencies, there is no interference between them because the receiving antennas are highly directional.

To pick up these signals, you need to rotate your dish antenna. You can equip your antenna with a remote control motor to rotate the dish from inside your house. Other satellite systems provide electronic switches that correctly position the antenna.

Earth transmitters send TV signals to satellites at 1,000 watts. This is called the uplink of the transmission. The satellite receives the signal, usually at around 6 gigahertz in frequency, cleans it up, and retransmits it back over the United States (see the schematic in Fig. 1).

The return signal arrives at between 3.7 and 4.2 gigahertz and at a power of 5 watts. Because the signal is so weak, you need a large receiving dish to gather and concentrate the signal (see the Photo). For most of North America, a 10-foot dish is the minimum required for a good picture. The dish gathers and concentrates the signal, and bounces it back into the feed assembly, a combination receiver/amplifier suspended opposite the center of the dish (see Fig. 2). The feed assembly directs signals into the receiving amplifier, called a low-noise amplifier, or LNA.

The LNA amplifies the signal and sends it to the downconverter, also part of the feed assembly. The downconverter changes the high-frequency microwave signal to a lower, more manageable signal of about 70 MHz. It then sends the signal along a standard TV cable wire (Radio Shack part number RG-59U) into your home.

Finally, the signal travels indoors to the receiver, letting you select a channel from 1 to 24. You receive the signal through a radio frequency signal (usually on channels 2, 3, or 4) connected to the antenna input of any television. Direct video/audio outputs are included for connection to a video tape recorder or a large-screen projection TV.

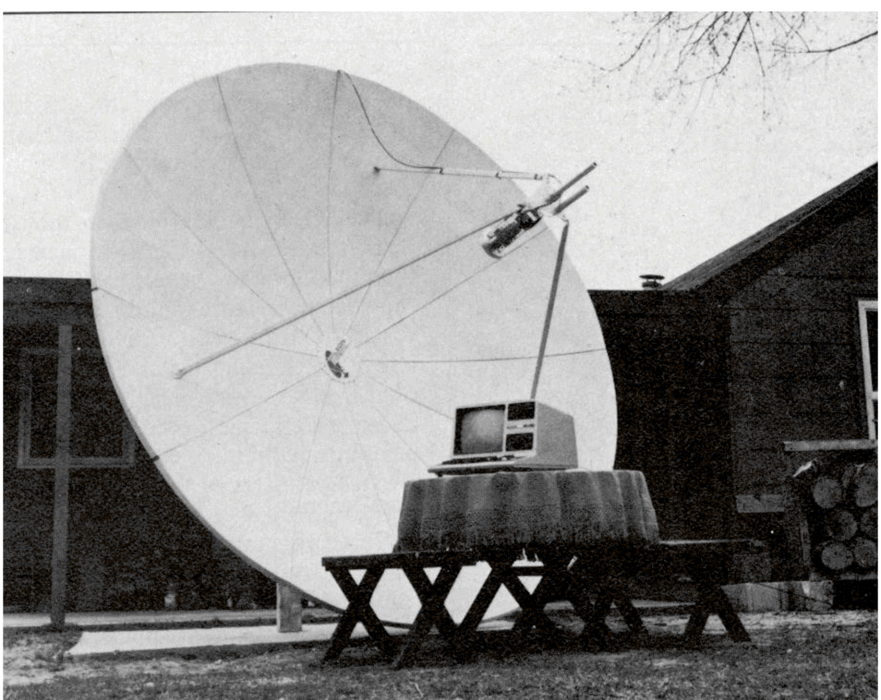


Photo. One of the authors' microwave antennas.

Broadcast Scrambling

Potential satellite owners need not worry about spending money for a receiving system that only picks up scrambled signals. As of this writing, scrambling

Program Listing. Satellite guide and site survey program.

```
10 CLS:PRINT@512,"ENTER YOUR CHOICE <S>ITE SURVEY <P>ROGRAM GUIDE
";:GOSUB640:IFIK$="S"THENGOTO1000ELSEIFIK$<>"P"THENGOTO10
20 CLEAR1000:DIM A$(24),S(75):GOSUB580
30 DATA Nickelodeon/Arts,PTL,WGN,Spotlight,The Movie Channel,WTBS,
E.S.P.N. Entertainment & Sports Programming Net,A.E.T.N./Christian
 Broadcasting Network,USA Cable Network/C-Span,Showtime,MTV (Music
 Television),Showtime
40 DATAHome Box Office,Cable News Network,Cable News Network Headl
ine News
50 DATA HTN Plus/ASCN/National Jewish Television,Cable Health Netw
ork,Reuters Monitor Service/Eternal Word Television,C-Span,Cinemax
,The Weather Channel,Modern Satellite Network/Daytime/HBO/USA Net.
,Cinemax,HBO
60 FORZ=1TO24:READA$(Z):NEXT:CLS
80 PRINT@87,CHR$(176);CHR$(176);STRING$(9,188);CHR$(176);CHR$(176)
90 PRINT@148,CHR$(184);CHR$(190);STRING$(15,191);CHR$(189);CHR$(18
0);CHR$(144)
100 PRINT@210,CHR$(184);STRING$(21,191);CHR$(189);CHR$(144)
110 PRINT@273,CHR$(186);STRING$(24,191);CHR$(144)
120 PRINT@337,STRING$(25,191);CHR$(149)
130 PRINT@401,CHR$(138);STRING$(24,191)
140 PRINT@466,CHR$(139);CHR$(175);STRING$(20,191);CHR$(143);CHR$(1
29)
150 PRINT@532,CHR$(131);CHR$(143);STRING$(15,191);CHR$(143);CHR$(1
35);CHR$(129)
160 PRINT@599,CHR$(130);CHR$(131);CHR$(131);STRING$(7,143);CHR$(13
1);CHR$(131);CHR$(129)
170 PRINT@365,"Earth";
180 REM  PLOT SATELLITES
190 SET(0,24):PRINT@513,"Satcom 3";:SET(2,28):PRINT@578,"Comstar D
4";
200 SET(5,33):PRINT@707,"Westar 5";:SET(11,38):PRINT@774,"Satcom 2
";
210 SET(18,41):PRINT@842,"Anik A3/A2";:SET(28,44):PRINT@976,"Anik
D1";
220 SET(47,46):PRINT@911,"Anik B";
230 SET(64,46):PRINT@993,"Westar 4";:SET(78,43):PRINT@937,"Comstar
 D1/D2";
240 SET(87,40):PRINT@878,"Westar 3";:SET(0,21):PRINT@449,"Satcom F
1";
241 SET(0,18):PRINT@385,"Satcom F5";:SET(96,37):PRINT@817,"Comstar
 D3";:REM F5 & D3
242 SET(101,34):PRINT@756,"Satcom F4";:SET(105,31):PRINT@694,"West
ar 1/2";:REM F4 & WEST 1/2
250 S=1:REM  STAR TWINKLING
260 GOSUB550:X=0
270 FORA=1TO75:B=RND(1024):S=15359+B:C=PEEK(S):IFC>32THENNEXTELSEP
OKES,46:S(X)=S:X=X+1:NEXT
280 TW=RND(X):POKES(TW),32:FORDE=1TO20:IFINKEY$=""THENNEXT:POKES(T
W),46:GOTO280
290 REM  SATELLITE DATA VIDEO PAGE
300 CLS:GOSUB550
310 PRINT@128,"SATELLITE","POSITION","COUNTRY"
320 PRINT"----------------------------------------------------------
-------"
330 PRINT"Satcom F3R","131 degrees","U.S.A."
340 PRINT"Comstar D1","127 degrees","U.S.A."
350 PRINT"Westar 5","123 degrees","U.S.A."
360 PRINT"Satcom F2","119 degrees","U.S.A."
370 PRINT"Anik 3","114 degrees","Canada"
380 PRINT"Anik B","109 degrees","Canada"
390 PRINT"Westar 4"," 99 degrees","U.S.A."
400 PRINT"Comstar D2"," 95 degrees","U.S.A."
410 PRINT"Westar 3"," 91 degrees","U.S.A."
420 PRINT"Comstar D3"," 87 degrees","U.S.A."
430 PRINT"Satcom F4"," 83 degrees","U.S.A."
440 GOSUB610
450 REM  SATCOM DATA VIDEO PAGE
460 CLS:GOSUB550
470 PRINT:PRINT" Satcom F3R Programming Services For Transponders
1 Through 12"
480 X=194:FORP=1TO12:PRINT@X,P,A$(P):X=X+64:NEXT
490 GOSUB600:GOSUB610
500 CLS:GOSUB550
510 PRINT:PRINT" Satcom F3R Programming Services For Transponders
13 Through 24"
520 X=194:FORP=13TO24:PRINT@X,P,A$(P):X=X+64:NEXT
530 GOSUB600:GOSUB610
540 RUN
550 PRINT@7,"SATELLITE GUIDE - Hit any key for more information";
560 RETURN
580 REM
                MAKE BORDER

590 A$=STRING$(63,131)+CHR$(191):FORA=1TO12:A$=A$+CHR$(26)+CHR$(8)
+CHR$(191):NEXT:FORA=1TO13:B$=B$+CHR$(191)+CHR$(26)+CHR$(8):NEXT:B
$=B$+CHR$(191)+STRING$(62,176):RETURN
```

Listing continued

is almost nonexistent in the satellite industry. Only Home Box Office has threatened to modify their signal in the future.

Some programming on Canada's Anik satellite is scrambled, but it has technological problems that are so significant that subscribing cable companies have appealed to them to discontinue this practice.

The Latest Developments

Satellites launched since 1982 have more power than previous units. Consequently, picture quality remains high with less efficient (and less expensive) receiving systems.

Because of these advances, great differences exist between equipment performance and cost. You can find dish antennas for around $1,200 that give

"Satellites launched since 1982 have more power than previous units. Consequently, picture quality remains high with less efficient (and less expensive) receiving systems."

the same performance as those costing in excess of $10,000. You can install complete systems today for about $2,000 to $3,000.

The latest line of receivers on the market are more attractively packaged than those of a few years ago. They also provide new features and innovations.

Richly furnished cabinets, illuminated meters, and digital channel readouts enhance front panel appearance. Several companies even offer handheld, infrared remote control.

The Program

Our program contains a satellite guide option and a site survey option. The satellite guide option lists all satellites and their position relative to earth, and prints a graphics display. It also lists the 24 TV channels on the Satcom F3R satellite.

The site survey option prompts you for your location and tells you if you

✓ See List of Advertisers on Page 291

Listing continued

```
600 PRINT@128,A$;:PRINT@128,B$;:POKE16383,191:RETURN
610 GOSUB550:FORX=1TO200:IFINKEY$=""THENNEXTELSERETURN
620 PRINT@25,"   ";:FORX=1TO50:IFINKEY$=""THENNEXT
630 GOTO610
640 IK$=INKEY$:IFIK$=""THENGOTO640ELSERETURN
1000 CLEAR100:DEFDBL A-Z
1010 PI=3.141592
1020 R=3963
1030 H=22300
1040 CLS:PRINT@18,"SATELLITE SITE SURVEY PROGRAM":PRINTSTRING$(64,
"-")
1050 PRINT"ENTER THE LONGITUDE OF THE SATELLITE.  HOW MANY DEGREES
 ";
1060 INPUT SD:INPUT"HOW MANY MINUTES ";SM
1070 INPUT"IS IT WEST OR EAST LONGITUDE (ENTER W OR E) ";SB$
1080 SA=SD+SM/60:IF SB$="E"THEN SA=-SA
1090 INPUT"ENTER DISH (ANTENNA SITE) LONGITUDE.  HOW MANY DEGREES
";DD
1100 INPUT"HOW MANY MINUTES ";DM
1110 INPUT"IS IT WEST OR EAST LONGITUDE (ENTER W OR E) ";DB$
1120 DA=DD+DM/60:IF DB$="E"THEN DA=-DA
1130 INPUT"ENTER DISH LATITUDE.  HOW MANY DEGREES ";TD
1140 INPUT"HOW MANY MINUTES ";TM
1150 TA=TD+TM/60:TA=90-TA
1160 A=ABS((SA-DA)*PI/180)
1170 C=TA*PI/180
1180 CA=SIN(C)*COS(A):TA=SQR(1/(CA*CA)-1)
1190 AA=ATN(TA)
1200 BS=SIN(A)/SIN(AA)
1210 TB=1/SQR(1/(BS*BS)-1)
1220 BB=ATN(TB):BB=BB*180/PI
1230 IF SA>DA THEN TR=180+BB ELSE TR=180-BB
1240 PRINT"THE TRUE BEARING (ANTENNA HEADING) IS ";
1250 PRINTUSING"###.##";TR
1260 X=SQR(R*R+(R+H)*(R+H)-2*R*(R+H)*COS(AA))
1270 SE=(R+H)*SIN(AA)/X
1280 TE=1/SQR(1/(SE*SE)-1):EL=ATN(TE)
1290 EL=EL*180/PI
1300 EL=90-EL
1310 PRINT"THE ANGLE OF ELEVATION IS ";
1320 PRINTUSING"###.##";EL;:PRINT" DEGREES"
1330 PRINT:INPUT"HOW MANY DEGREES IS THE MAGNETIC DEVIATION OF YOU
R AREA ";MD
1340 INPUT"IS THAT WEST OR EAST (ENTER W OR E) ";A$
1350 IFA$="W"THEN MD=MD+TR ELSE MD=TR-MD
1360 PRINT"YOUR COMPASS HEADING SHOULD ";USING"###.##";MD;:PRINT"
DEGREES"
1370 INPUT"HIT <ENTER> FOR MENU ";OL$:RUN
```

can place a satellite receiving station on your property by calculating the heading and elevation of a dish for any given satellite.

The program first prompts you with a menu containing the two options. Hit the S key to perform a site survey or the P key for a satellite programming guide.

The satellite guide section presents a visual picture of the earth, with current satellites in their respective positions. The program displays information about the various satellites, including their longitude (true position) and their country of origin.

There must be an unobstructed line from your dish to the satellite. In the winter, trees have no effect on microwaves, but the addition of water in foliage in the spring and summer wipes out reception.

The mathematics used in the program involve spherical trigonometry and is beyond the scope of this article.

Data Input

It's easy to accurately determine the latitude and longitude of your house by checking a map or referring to the deed to your house. You must also input the magnetic deviation (the difference between true north and magnetic north) in your area. You can get this information from your local airport or, if you live along the coast, from the local Coast Guard base.

Enter the longitude of the satellite you wish to find by copying the information from the satellite guide section in the program. Enter fractional degrees using either a decimal point or by answering the prompt for the number of minutes of arc. For example, if your latitude is 74 degrees, 50 minutes, you can enter this into the program in two ways:

```
ENTER DISH (ANTENNA) LONGITUDE
HOW MANY DEGREES? 74.8
HOW MANY MINUTES? 0
```

or

```
ENTER DISH (ANTENNA) LONGITUDE
HOW MANY DEGREES? 74
HOW MANY MINUTES? 50
```

Installing It Yourself

If you're interested in purchasing your own satellite receiver, we suggest you talk to your local satellite dealer. Unless you're mechanically inclined, a site inspection for reception quality, and siting the antenna should be done by a professional.

With a distance of 22,300 miles from satellite to the equator, an antenna situated off by 1 degree will miss its intended satellite by 1,000 miles. A preliminary inspection will save you a lot of wasted effort in an area of poor reception.

By running your own cable, pouring the concrete, and assembling the equipment yourself, however, you can save hundreds of dollars.

We purchased a Wilson Microwave system. The Wilson system is not a turnkey system; that is, the manufacturer does not supply all the material necessary to put the system into operation, but only a few connectors and some wire aren't included.

The wire is standard multiconductor cable. It runs the power supplied by the indoor receiver out to the low-noise amplifier and downconverter. You also need an RG-59U wire to bring the TV signal from the downconverter into the house. You can purchase both types of wire at Radio Shack.

One person cannot install this system alone. Laying the concrete foundation for the dish and assembling its panels are two-man operations.

An 11-foot dish has a surface area of 95 square feet. On a 100-square-foot area, an 80 mph wind exerts a 2½ ton force. This kind of stress makes a secure base essential.

Conclusion

In retrospect, the expense we incurred to construct a working earth satellite receiver system was higher than originally estimated. Several nuts and bolts, two special microwave connectors (available from a satellite system dealer), and wire and cables all were additional expenses.

The system is fairly easy to operate, although you have to adjust the fine tuning every time you change stations.

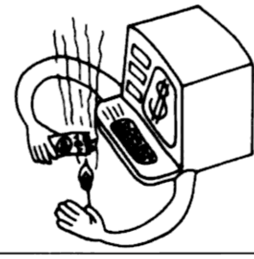
During heavy rain, we pick up some snow on the picture, but other than that, picture quality is good.

What's on TV tonight? Everything! ■

Dan Keen and Dave Dischert can be reached at Soft Horizons Computer Software, RD1 Box 432, State Highway 83, Cape May Court House, NJ 08210.

DOES YOUR COMPUTER WA$TE MONEY?

It Does If Your Software Wastes Valuable Time



Our programs and systems are the result of experience gained working with custom systems. They were developed for the general user with the user's needs in mind at all times. Every effort has been made to make the very best possible use of the hardware in terms of speed, capacity, and efficiency.

All versions of mailing list and data bank are programmed in compiler basic using 'ISAM' files to eliminate the need for time consuming, inefficient, and expensive sorting. Through the use of 'ISAM' files, the maximum number of records that can be handled by the computer is only limited by the availability of disk space and not by the amount of computer memory needed for sorting and the complete file is accessible at any time.

The use of 'ISAM' files is not the only time saver, as compiler basic programs run considerably faster than ordinary basic programs. For example: the loan amortization system was programmed in compiler basic just to speed up the time it takes to calculate the loan payments and produce the amortization report.

Both versions of our inventory system are programmed in interpreter basic, utilizing both direct access and variable length records to maximize access speed and the use of available disk space.

The dedication, ability, and years of experience of both the system analyst and the programmer combine to make these programs and systems among the finest that you have had the opportunity to use and certainly one of the best values on the market.

DATA BANK SYSTEM

Builds and maintains user defined data file(s) with up to 16 (12 for model III) user defined data fields. The user specifies file name, control field data, data field names, and data field size. The system adds records, changes records, deletes records, displays individual records, prints all records in user defined format with user's choice of fields and order, and prints records by user defined search as to field and value in user defined format with user's choice of fields and order.

Model 2, 12 ..... $295.00
Model III $275.00

MAILING LIST SYSTEM (Alpha Sequence)

Builds and maintains mailing list in alpha sequence by entire name. The system adds records and checks for duplicates, changes records, deletes records, displays individual records, prints all records, prints records by user specified zip code search, prints all mailing labels, prints individual labels (as many as requested), prints labels by user specified zip code search, & prints telephone directory. The data file created by this system can be used as input for the mailing list (zip code sequence) system.

Model 2, 12 ..... $200.00
Model III $175.00

MAILING LIST SYSTEM (Zip Code Sequence)

Builds and maintains mailing list in alpha sequence within zip code sequence. This system adds records and checks for duplicates, changes records, deletes records, displays individual records, prints all records in alpha sequence within zip code, prints records by user specified zip code, prints all mailing labels in alpha sequence within zip code, prints individual labels (as many as requested), prints labels for user specified individual zip codes, prints telephone directory in alpha sequence within zip code, and reads in file(s) created by mailing list (alpha sequence).

Model 2, 12 ..... $250.00
Model III $225.00

LOAN AMORTIZATION

Calculates personalized loan payments and interest. User specifies loan amount, interest rate, and length of loan. Displays monthly payment, displays monthly analysis, displays totals, prints monthly payments, prints monthly analysis, and allows user to enter over ride of monthly payments and recalculate totals and print or display monthly analysis.

Model 2, 12 ..... $125.00
Model III $115.00

INVENTORY CONTROL SYSTEM

(Retail, Wholesale, Manufacturer)

Builds and maintains records on all in stock items. Records contain user's part number, manufacturer's part number, on order quantity, in stock quantity, reorder point, wholesale price, retail price, sales history for preceeding month, year, same month last year, this year to date, last year to date. Displays individual item record, prints complete record for all items, and prints suggested purchase order. The add routine checks for duplicates.

Model 2, 12 ..... $275.00
Model III $225.00

DESK CALENDAR

Each month prints on 14⅞ x 11 paper. Each day is blocked to allow appointments entry. User specifies beginning month and number of months and number of calendars to be printed. 4 lines of user information may be printed on each month which makes it useful as an advertising media.

Model 2, 12 ...... $49.95
Model III $49.95

WALL CALENDAR

Each year prints on 8½ x 11 paper. User specifies beginning month and number of calendars to be printed. 4 lines of user information may be printed on each month which makes it useful as an advertising media.

Model 2, 12 ...... $49.95
Model III $49.95

For Additional Information Send for User Documentation — $15.00 per copy.

Minimum Hardware Requirements
Model III .... 48k 1 Disk
Model 2, 12 .... 64k 1 Disk

- Shipping Costs Included on Pre Paid Orders.
  N.C. Residents Add 4½% Sales Tax.
  Personal Checks Allow 3 weeks.

VISA

CDP

MasterCard

Route 3, Box 1 • Hamptonville, N.C. 27020
(919) 468-2113
8 am - 5 pm Eastern

✓ 65

A Modem Sampler

by R.A. Langevin

★★★★

J-Cat
$149

★★★★

103/212 Smart Cat
Novation Inc.
18664 Oxnard St.
Tarzana, CA 91356
$595

★★★½

212A/D Direct Connect Modem
Universal Data Systems
5000 Bradford Drive
Huntsville, AL 35605
$745

Choose one of these modems and get the right communication ability for your applications.

These three direct-connect modems offer attractive alternatives to meet the need to transmit information from your computer over telephone lines. They differ significantly in price and capabilities, but one of them is likely to suit your needs.

Each of the three modems works well and does all that it's supposed to. If you can get along with a 300 baud modem, the J-Cat is hard to beat in terms of price and performance. It's small, works perfectly, and is about as inexpensive a modem as you can buy.

If you need a 1,200 baud modem, in addition to or instead of 300 baud operation, the Smart Cat offers a near-perfect alternative. It's remarkably small, offers all the features you're likely to need, and is competitively priced.

Alternatively, if you want a modem that can handle any 300 or 1,200 baud operation you might encounter, the UDS 212A/D is an excellent choice. It offers all the flexibility you're likely to need, is easy to operate, and includes diagnostic capabilities to isolate communications problems on the line, in the modem, or in the terminal equipment. All in all, it's a good product for commercial or industrial applications.

Table 1 summarizes major features. Which modem you choose depends on your finances and applications.

The three modems differ radically in size. The Novation J-Cat is scarcely larger than a pack of cigarettes. Its more capable companion, the 103/212 Smart Cat, resembles a tall paperback book. The UDC 212A/D, clearly intended for commercial use, has a footprint that is almost 3/4 of a square foot!

The J-Cat and UDC units are housed in plastic cases and the Smart Cat comes in a nonmagnetic metal case. All the units should retain their physical integrity in any ordinary use.

A conventional plug-in wall transformer powers the two Novation modems. The UDC unit has an integral alternating current (ac) power supply.

Neither of the Novation units has an on/off power switch, an unfortunate omission. The power requirements of these modems are nominal, however, ranging from 7–13.5 watts, so it's feasible to leave them turned on all the time.

J-Cat

The J-Cat is a simple, full duplex, 300 baud modem that has surprising capa-

Feature	J-CAT	103/212 Smart CAT	UDS 212A/D
Size (Inches)	5.0 x 1.9 x 1.3	10.0 x 4.7 x 1.2	10.85 x 9.76 x 2.42
Baud Rates	300	300/1200	300/1200
Auto Dial	See Text	Yes	Yes
Manual Answer	Yes	Yes	Yes
Auto Answer	Yes	Yes	Yes
Keyboard Dial	No	Yes	Yes
Manual Redial	Yes	Yes	Yes
Auto Redial	No	Yes	Yes
Dial Modes	See Text	Pulse/MFTD	Pulse/MFTD
Memory	None	Last #	5 #'s + Last #
Asynchronous	Yes	Yes	Yes
Synchronous	No	No	Yes
Resettable Defaults	No	Yes	No
Self Test	No	Yes	Yes
Analog Loopback	Yes	Yes	Yes
Local Digital Loopback	No	Yes	Yes
Remote Digital Loopback	No	Yes	Yes
Phone Jack(s)	RJ11C	RJ11C	RJ11C/RJ45S
Power (Watts)	8.0	13.5	7.0
Power Supply	Wall XFMR	Wall XFMR	Internal
Power Switch	No	No	Yes
Status Response	LED	DISPLAY	DISPLAY
Phone Service	1*	1*	1, 2, 3*
Cables Provided	All	Phone Line	Phone Line
Price	145.00	595.00	745.00

*Phone Service: 1 - Permissive, 2 - Programmable, 3 - Private Line

Table 1. Summary of modem characteristics.

bility in spite of its tiny size. You operate it with two push buttons; it has two LEDs that indicate its status. One push button puts the modem on line; the second disconnects it. Pressing the two buttons simultaneously puts the modem in analog loopback mode, so that its output connects to its input—an invaluable test to verify its operation.

You select auto answer or manual answer modes with a slide switch on the back of the unit. The two LEDs indicate when the modem is off-hook and ready for operation.

Although the J-Cat is not directly usable in auto-dial mode, you can implement auto pulse dialing with dial tone detect if 1 input and 1 output bit are available from another port. The operations manual provides a simple program to implement directory dialing and automatic redial using these bits. The program is written in Applesoft Basic but you can readily adapt it to Microsoft Basic.

The J-Cat is relatively unique among modems in providing automatic mode selection. When on line, the modem alternates between originate mode for two seconds and answer mode for four. As soon as it detects a carrier, the modem locks in its current mode. This relieves the user of any concern for selecting the modem's operating mode—a very useful feature.

The J-Cat comes with all the required cables, including one terminated with a female RS-232C D-connector that plugs into the output of your serial interface. A Y cable is also provided; one end of it plugs into the phone line via an RJ11C plug. The other leg of the Y terminates in a socket that accepts the RJ11C plug on the telephone.

You don't need to buy anything else. Simply hook up the modem according to the clear directions in the operations manual, load your terminal program, and you're ready to communicate.

The J-Cat's operating manual is a 4¼- by 9½-inch booklet of 21 pages that is clearly written and provides all the necessary information to install and operate the modem. A schematic is included, with detailed pin-outs for the connectors on the back of the unit.

103/212 Smart Cat

The 103/212 Smart Cat is the J-Cat's big brother. In addition to operation at both 300 and 1,200 baud, the user-programmable Smart Cat provides numerous operating conveniences.

In normal operation this modem has no visible controls. You select all operating modes by entering commands from the terminal or computer keyboard. As a consequence, you can tuck the modem away from the operating position in the most convenient location. You can even place it inside a piece of equipment if the thermal environment isn't severe.

The Smart Cat has five option switches located behind its front panel. These are readily accessible if you pry the panel off with a small screwdriver. Once you set them, these switches will not normally need resetting.

Switch 1 sets the command mode. It determines the character you must send from the terminal to indicate to the modem that a command follows. Normally, the percent sign is the command indicator. By using the option switch, you can select any other character, including a control character, and substitute it for the percent sign. You terminate all commands with a carriage return.

Switch 2 sets the response mode and gives the user the option of receiving modem responses on the terminal screen in full English words or in the terse mode as single characters.

Switch 3 sets the data rate to which the modem is automatically set when you turn on the system. Naturally, you can change the rate afterward from the terminal keyboard.

Switch 4 enables or disables the auto answer mode. You cannot subsequently reset this switch selection from the keyboard.

Switch 5 determines whether the modem senses the Data Set Ready (DSR) and Clear To Send (CTS) lines all the time or only after detection of a carrier on the phone line.

You maintain normal control of the Smart Cat with 21 commands that you can enter directly from the terminal or computer keyboard. Depending on the type of command, some have arguments and/or default values.

Initialize puts the modem in its normal working state, while Hangup disables the modem, waits three seconds, and hangs up the phone. You cannot receive data during the three-second wait.

Dial, followed by a string of up to 32 characters, takes the phone line off hook and dials in accord with the character string contents. Allowable characters comprise 0–9, #, *, I, P, and W. I indicates that the modem generates telephone pulses rather than tones until the next P or W.

P forces a wait for a dial tone; if no dial tone occurs in five seconds, the modem hangs up and gives a No Dial response. W forces a five-second wait before the modem begins dialing.

You can use another dialing command, Count, followed by an integer, to determine how many rings the modem waits for before aborting a call. Redial dials the last number at intervals of 40 seconds until you reach the number or you've dialed it 10 times.

Voice puts the modem in voice mode, and Modem puts it in data mode. Pickup puts the phone line off hook, and also puts the modem in voice mode. Answer and Originate put the Smart Cat in answer or originate mode, respectively; they both take the phone line off hook. You must follow the last mode command, Giveback, with zero or 1 to put the modem in full or half duplex mode 1, respectively.

Speed, followed by 1 or 2, sets the

✓ See List of Advertisers on Page 291

modem speed to 300 or 1,200 baud. In answer mode, the modem automatically sets itself to the incoming data's speed, irrespective of speed selection.

Follow New with a single ASCII character to substitute that character for the current command character. Use Echo followed by zero or 1 (with a default of zero) to enable (zero) or disable (1) the echo of commands back to the terminal. The modem does not echo data.

Query forces the modem to return a single character that describes its status. Long, followed by zero or 1, causes the modem to provide full English (zero) or single-character (1) responses. Break, followed by an integer n, sends a space on the phone line. The length of the space is n times 250 milliseconds.

With the modem in voice mode, XMIT followed by a phone number transmits the DTMF tone pairs corresponding to the given number.

Follow Unlisten with a one- to four-character hexadecimal (hex) argument to transmit the number of bytes given by the hex argument as data. This permits transmission of data that the modem would otherwise interpret as command strings.

Format followed by a single-character argument establishes the number of data and stop bits, and the parity of data transmitted over the phone line. The allowable arguments and their interpretations are shown in Table 2.

Test, followed by an integer from zero to 3, selects, respectively, hardware integrity, analog loopback, remote digital loopback, or local digital loopback test modes. The hardware integrity test also occurs automatically when you turn on the machine. It verifies the modem's operation and correct connection of the RS-232C cable.

The responses the Smart Cat provides are clear and easy to understand. Ring indicates that the modem detects a ringback tone, and Busy indicates a busy signal. No Dial, CONN Lost, and No ANS are self-explanatory, although the No ANS occurs only after the specified number of rings.

UNSUCC means that for some unspecified reason the modem cannot complete the call. Ready means that carrier is acquired and communication established. Ring In tells you the modem detects an incoming ring. Finally, OK tells you that your last command is complete and the modem is ready for another.

While this description sounds complicated at first reading, operating the Smart Cat becomes second nature with a little use. The operations manual, identical in size to that of the J-Cat, is only slightly longer, yet it provides a fully adequate description of all the modem's operational and test features.

The Smart Cat package includes the wall mount power transformer and a cable to connect the modem to an RJ11C phone jack. An RJ11C socket on the back of the modem lets you plug the telephone directly into the unit.

The manufacturer provides no RS-232C cable, so you have to make one up yourself or buy one ready-made. The manual describes the necessary pin-out details at the cable's modem end. You have to determine the corresponding pin-outs at the serial connector on your terminal or computer.

Argument	Stop Bits	Data Bits	Parity
0	1	7	Mark
1	1	7	Space
2	1	7	Odd
3	1	7	Even
4	1	8	None
8	2	7	Mark
9	2	7	Space
A	2	7	Odd
B	2	7	Even
C	2	8	None

Table 2. Format allowable arguments and interpretations for the Smart Cat.

UDS 212A/D

The UDS 212A/D includes most of the features available in the Smart Cat as well as a number of others that make it a natural selection for system houses that want to carry only a single modem that can be configured to meet most communications needs.

Additional features available in the 212A/D include selectable permissive or programmable transmit levels, operation with private phone lines, optional synchronous operation at 1,200 baud, battery-backed memory for five 30-character dial strings, and capability to wait for a second dial tone. The modem also displays and edits stored dial strings, and has a Help command to display and describe commands.

The 212A/D is built on two circuit boards. The top board contains the modem circuitry, and the bottom the automatic calling unit (ACU) and the power supply. DIP (dual in-line package) switches on the top board permit setting a variety of options.

These include forced answer, forced originate modes, and private line operation. You can disconnect on long receive or transmit space, loss of Data Terminal Ready (DTR), or loss of carrier. The 212A/D also has auto answer always or only when DTR is enabled, synchronous or asynchronous operation, and 9-, 10-, or 11-bit characters (including stop bits).

Strapping options on the board permit grounding or ungrounding the modem chassis; selecting internal, external, or slave transmit clock options; and selecting permissive or programmable transmit levels. A DIP switch on the lower board enables or disables the automatic calling unit.

In operation, you should think of these option selections as permanent, since you must remove the modem cover to change them—not something you care to do often.

The 212A/D starts up in initialization mode. It expects to receive an uppercase, two-character string, EN, from the terminal or computer. The modem uses this input to establish the operational character rate and data format. After it receives EN, the modem responds with:

```
(CR) (LF)
UDS 212 DIALER
(CR) (LF)
:
```

The colon is the prompt from the ACU and indicates that the unit is ready to receive commands. Eleven commands, input as single, uppercase letters, are available to control the operation of the ACU.

H, the Help command, displays a brief summary of the available commands on the terminal or computer screen.

D lets you dial a telephone number directly from the keyboard. You can specify a number of up to 30 characters long containing both digits and operators. Five operators are available: W, D, E, A#, and space. W in the dial string causes the ACU to wait for a second dial tone. This important feature permits use of the modem with alternative carriers such as MCI, Sprint, and so on.

D in the dial string introduces a pause of 1.5, 3.0, or 4.5 seconds in the dialing operation. You select the length of the pause. E at the end of the dial string immediately terminates the calling sequence and connects the telephone to the phone line for voice communication.

A#, after all other digit entries and where # is a digit from 1 to 5, takes the dial string from the memory register identified by # to complete the original dial string. The space character is purely a cosmetic operator to improve the string's readability.

ACU command L, followed by a digit from 1 to 5, loads any of the five memory registers with a dial string. The ACU repeats the loaded string back to the CRT and returns to command mode with the display of the : prompt. P, the Print command, displays the contents of all loaded memory registers on the screen.

C, followed by a digit from 1 to 5, clears the contents of the correspondingly numbered memory register. If C is followed by the character L rather than a digit, all memory registers clear. Where # is a digit from 1 to 5, # causes the ACU to dial the dial string in the memory register identified by #.

R redials the last number dialed, and X redials the last used dial string continually until the modem detects an Answer Back Tone.

Q, the Quit command, aborts any ACU operation. B takes the phone line off hook, and N cancels the B command and returns the phone line to on hook status.

A primitive, although entirely adequate, editing capability lets you correct mistakes made when you enter dial strings from the keyboard. Enter the character @ to delete the last character entered. If you enter several @ characters, the program deletes the corresponding number of previous keyboard entries.

The 212A/D provides simple English responses to the terminal or computer screen to indicate the status and disposition of ACU actions. These are self-explanatory and include such responses as Off Hook when you initiate dialing, No D.T if the modem detects no dial tone, echo of the number dialed, No ABT if the modem receives no Answer Back Tone, and Busy, Complete, or Abort to indicate call disposition.

In addition to the options you set on the modem board itself, you can set six more options from the keyboard. OA#, where # is zero or 1, requires dial tone detection before dialing (zero) or waits four seconds and then dials (1) even though no dial tone is present. Zero is the default.

OB# provides pulse dialing, the default, where # is zero and multifrequency tone dialing where # is 1. OC# provides DSR active if # is zero, the default, or inactive if # is 1.

OD# sets the wait time before an unsuccessful dialing attempt aborts. Values of # from zero to 4 give wait times from 15 to 75 seconds. The default is 30 seconds. OE# sets the delay produced by the D command in a dial string. Values of # from zero to 2 give delays of 1.5 to 4.5 seconds. Three seconds is the default.

OF# lets you turn on or off the echo of commands back to the screen. A value of zero for # turns off the echo and 1 turns it on. Echo On is the default. OG0 reinitializes the modem to its start-up condition and requires entry of the initialization sequence EN to enable the automatic calling unit.

While it is convenient to set some of the modem parameters with the OA–OF commands, whatever settings you've established are lost if you turn off the modem or reinitialize it by using OG0.

Finally, six buttons on the modem's front panel provide the last remaining elements of control. Four of these activate a modem self-test or the three loopback test modes. A fifth forces the modem into 1,200 baud mode, and the sixth puts the telephone set on or off line.

Overview

All the modems are covered by a one year warranty on parts and labor. None of them is readily repairable by the user, nor is this recommended.

You must return the two Novation modems to the factory for repair. You can have the UDS modem repaired at the factory or at a number of authorized service locations. Since none required service as I prepared the review, I don't have information on the turnaround time for repair.

Reliability of the Novation units should be very good since they use a high degree of integration—the J-Cat has four integrated circuits and the Smart Cat has only a few more.

The UDS 212A/D might be very reliable as well, but the extraordinary parts count is a potential source of worry—the modem board alone has 83 integrated circuits!

None of the three modems comes with communications software. I operated all of them for test purposes with the Lobo Max-80 using both the COM-M/CMD terminal program that is integral with LDOS and SMODEM, a recent version of the widely available, public domain MODEM7, operating under CP/M 2.2.

I accessed CompuServe and several local bulletin board systems at both 300 and 1,200 baud (only 300 baud for the J-Cat) to verify the modems' operational features. Each performed exactly as represented in the respective manuals, and I encountered no problems of any kind.

Most of my complaints are minor. The J-Cat's remarkably small size makes you wish you could tuck it inside a terminal or a computer case. Unfortunately, you must use the unit's push buttons to put it on line and back off line. It's also aesthetically annoying that the label on the face of the modem is printed upside down.

The Smart Cat leaves little room for complaint. It does what it's supposed to do and does it well, although it would be nice to have a power switch.

The UDC modem has some problems. As I mentioned, you must take the cover off to set the character size. This is a minor annoyance that becomes a major one when you remove the cover and the front panel falls off. It turns out to be devilishly difficult to get back in place.

A more serious problem exists with the telephone set. Although you can plug the phone into the back of the modem and activate it with the talk switch on the front panel, you can't use the phone at all unless the modem is turned on.

It is also unfortunate that, although you can reset a number of modem parameters from the keyboard, you can't reset the defaults for these parameters and you are returned to the built-in defaults whenever you turn off or reinitialize the modem.

The UDS manual deserves special mention. Unlike the manuals for the Novation units, the UDS manual tells you both too much and too little. It is full of acronyms, usually defined long after they are first introduced if they are defined at all.

It tersely describes the bewildering array of available options that you can set or strap on the modem board, but you get no guidance as to what options you should select in what circumstances. An entire chapter is devoted to "theory of operation," but after reading it you know nothing useful about the modem.

Last, although the battery-backed memory for dial strings is a convenience and works perfectly, I can't understand why, with the low price of CMOS memory, storage is provided for only five numbers. Except in dedicated service, five numbers don't seem to be enough. ■

Contact R.A. Langevin at 7621 Fontaine St., Potomac, MD 20854.

ACCESS UNLIMITED

Merry Christmas Radio Shack™ Users!!!!!

PERCOM DATA™ DISK DRIVE HOLIDAY *SPECIALS:*

You always expect highest quality from Percom Data™. Each Drive is thoroughly tested and gets a 48 hour burn-in so you get trouble free performance!

★ MODEL I™

DDA-35	Upgrade to double density	**$ 99.95**
TFD40-1	Single sided/double density — one drive	**$239.00**
TFD40-2	Single sided/double density — two drives	**$449.00**
AFD40-1	Single sided/double density — one drive add-on	**$239.00**
AFD40-2	Single sided/double density — two drive add-on	**$449.00**

CLOSE-OUT! Limited Quantities

TFD100-1	35 track single sided/double density	**$239.00**
TFD42-1	40 track flippy-one drive	**$269.00**
TFD42-2	40 track flippy-two drive	**$499.00**
AFD42-1	40 track flippy add-on	**$269.00**

★ MODEL III™

TFD340N1	Single sided/double density — one drive	**$359.00**
TFD340N2	Single sided/double density — two drives	**$559.00**
ADD340N1	Single sided/double density internal add-on	**$199.00**
ADD340X1	Single sided/double density external add-on	**$229.00**
TFD344N1	Double sided/double density — one drive	**$419.00**
TFD344N2	Double sided/double density — 2 drives	**$699.00**
ADD344N1	Double sided/double density internal add-on	**$289.00**
ADD344X1	Double sided/double density external add-on	**$319.00**

CLOSE-OUT! Limited Quantities

TFD342-1	40 track flippy — one drive	**$339.00**
TFD342-2	40 track flippy — two drives	**$569.00**
ADD342X1	40 track flippy external add-on	**$299.00**

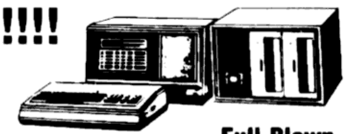
BIG DISCOUNTS ON PRINTERS

NEW! Brother CX15™	Only **$ 559.00**
New! Brother HR1A™	Only **$ 899.00**
C.itoh F10™	Only **$1495.00**
Okidata™	All Models **Call for our super low prices!**
Star Micronics 10X & 15™	**Call for our super low prices!**
Star Micronics Gemini 10™	(CLOSEOUT) Only **$ 259.00**

DIABLO™ MODEL 2300 MATRIX PRINTER
Close Out Special —

• Rugged, commercial duty • 7x9 dot matrix • High Speed! 200 CPS! • Top of the line, highest quality
Reg. Retail $2495.00 ***FOR LIMITED TIME ONLY!***
WHILE QUANTITIES LAST............................ $999.00

A BUSINESS COMPUTER AT A PERSONAL COMPUTER PRICE!!!!!



iBEX 7202™

Full-Blown Business Computer/ Word Processor

★ **LAST CHANCE — WHILE QUANTITIES LAST!**

Regular Retail Over $10,000.00

NOW ONLY **$1879.00**

With FREE CP/M™ & MBasic 80 Ver. 5.2™

PLUS: Discounts on MicroPro™ Word Processing & ADS™ Accounting Software!

Main Features Of The iBEX 7202™:

- **Two 8 inch double-sided/double density floppy disk drives with 2.4 million characters total storage!** This is a super system and one you can afford.
- **64 KB RAM**
- **12″ High resolution, green screen video monitor**
- **Selectable 40, 80, or 132 character display**
- **Parallel & RS-232 interface**
- **Supports IBM 3740 format switchable**
- **Strong, compact, and styled for the office**

CALL FOR FURTHER INFORMATION

★ ***Christmas Sale! — 15% Off All Prices Listed***
PowerSOFT Software from Breeze/QSD Inc.™

One disk "BOOTS" up on either machine. At PowerSOFT, we don't believe in making a customer buy a separate version for Mod I or III. We ALWAYS include BOTH versions on the SAME disk for your maximum convenience. 80 Track versions are available on request. All titles are for Mod I or III unless specified.

PowerSOFT Product Title	Retail Price
SUPER UTILITY PLUS/3.0	$ 79.95
SUPER UTILITY PLUS/3.0 for MAX80	$ 99.95
INSIDE SU+/3.0 MANUAL	$ 19.95
SU+/3.0 TECH MANUAL	$ 14.95
INSIDE SU+/2.2z MANUAL	$ 19.95
SU+/2.2z TECH MANUAL	$ 14.95
"THE TOOLBOX" for LDOS	$ 69.95
"MASTER MECHANIC SET" for LDOS	$ 39.95
POWERDRIVERS for SuperScripsit™ (printer drivers):	
POWERDRIVER/E (EPSON MX-80/100)	$ 29.95
POWERDRIVER/P (PROWRITER)	$ 29.95
POWERDRIVER/F (F-10 STARWRITER)	$ 29.95
POWERMAIL	$ 99.95
POWERMAIL PLUS* — Mod I, III, MAX80	$150.00
POWERMAIL PLUS* — Mod II/12/16	$150.00
POWERDRAW	$ 39.95
POWERDOT (EPSON or PROWRITER Only)	$ 49.95
POWERTERM Smart Terminal Pkge	$ 29.95
DOSPLUS II Operating System For Model II, 12, and 16/Z80	$249.95
SCRIPLUS 3.0	$ 39.95
THE BASIC/S COMPILER SYSTEM	$ 49.95
MAKE/80 (Mod I or III)	$ 19.95
SUPER UTILITY (Mod I ONLY!)	$ 29.95
QUICK-FIX (Mod I ONLY!)	$ 19.95

**Coming out very soon. Please inquire for shipping date.*

Real World Control—Part II

by David L. Engelhardt

Use your Model III to control a burglar alarm, sprinkler system, or other household device. These programs work with the hardware in Part I.

This is the second of a two-part article for advanced builders to utilize a real-world interface and clock to operate a burglar alarm and sprinkler system with room for other real-world applications.

The four Assembly-language programs in this article let you implement home control devices through your Model III, a real-time clock, and the input/output (I/O) board described in Part I (November 1983, p. 216).

After you build and test your I/O board, you're ready to program your computer to take over some mundane household chores.

Clock Program

Program Listing 1 is almost identical to the one in "Real World, It's About Time" (*80 Micro*, March 1983, p. 342). I made a few slight changes to allow a relationship with the sprinkler and burglar alarm programs. The principal addition is the patch in lines 1270–1320. If activated, the sprinkler and burglar alarm programs patch themselves here.

These patches contain a jump address to the scan sections of the appropriate program. The clock's one-second interrupts trigger one-second scans of each program. If either or both of the programs become deactivated, they disengage themselves from the clock's patch region.

Burglar Alarm Program

Program Listing 2 is heavily commented. When executed, the program prompts you with specific questions regarding system status and locations where the alarm system is activated. I'll describe some of its basic functions below.

When you run the program, it indicates its status and asks for the code word allowing it to swap its state. The code word I used is Mom, but you can change it. The program shows each location where the alarm is on. This lets you decide which point to deactivate when you turn off the system.

An alarm system must allow time for you to exit and enter the building before the siren sounds. In this case there is a two-minute exit delay and a 30-second entrance delay. Two minutes after you leave, the system activates itself and starts scanning all of the activated alarm points. Upon entering, you must deactivate the system within 30 seconds or the

Program Listing 1. Clock Initialization.

```
00010 ;CLOCK INITIALIZATION AND INTERRUPT READ PROGRAM
00020 ;WRITTEN BY DAVE ENGELHARDT   3/15/82
00030 ;SAVE MEMORY SIZE OF -32447-    (THIS WILL ALLOW ROOM
00040 ;                               FOR PATCHING IN FUTURE
00050 ;                               PROGRAMS.)
00060 ;EXECUTE WITH A   /32448
00070 ;
00080 ;         INITIALIZATION SECTION
00090 ;
00100           ORG     7EC0H           ;INITIALIZATION SECTION
00110           DEFW    00H             ;SPARE
00120 BEGIN     DI                      ;DISABLE INTERRUPTS
00130           LD      A,12            ;USED TO SET INTERRUPTS
00140           LD      (4213H),A       ;ENABLE EXT INTERRRUPTS
00150           LD      HL,START        ;LOAD START OF CLOCK
00160           LD      (403EH),HL      ;  TO INTERRUPT VECTOR
00170           LD      A,16
00180           OUT     (0ECH),A        ;ENABLE TRS MAIN BUS
00190           LD      A,128           ;SET UP 8255 FOR
00200           OUT     (SETUP),A       ;  MODE 0
00210           XOR     A
00220           OUT     (OUTC),A        ;MAKE SURE PC7 IS OFF
00230           OUT     (0ECH),A        ;SHUT OF MAIN BUS
00240           IM      1               ;SET MODE 1 INTERRUPT
00250           EI                      ;ENABLE INTERRUPTS
00260 EXI       JP      1A19H           ;BACK TO BASIC
00270 ;
00280 ;
00290 INPUTA    EQU     30H       ;USED TO CONTROL CLOCK DATA LINES
00300 OUTB      EQU     31H       ;USED TO CONTROL CLK ADDRSS LINES
00310 OUTC      EQU     32H       ;USED TO CONTROL CLK FUNCTIONS
00320 SETUP     EQU     33H       ;USED TO CONFIGURE 8255
00330 ;
00340 ;         CLOCK INTERRUPT/READ PROGRAM
00350 ;
00360 START     PUSH    HL              ;SAVE REGISTORS
00370           PUSH    DE
00380           PUSH    BC
00390           PUSH    IX
00400           PUSH    IY
00410           PUSH    AF
00420           LD      A,16            ;VALUE FOR TRS BUS
```

Listing 1 continued

The Key Box

Model III
16K RAM Cassette Basic
32K RAM Disk Basic
Assembly Language
Editor/Assembler

siren sounds.

Upon re-entry to the program, you must enter the code word to deactivate the system. It allows three attempts to enter the code word correctly. In case of a code entry error, the program jumps back to Basic. I label user-changable parameters in the listings. Just remember to keep the time delay parameters in units of seconds.

Also, in regard to the time delays, note that I allow two points in Port 2 for entry time delays. I label these in the listing under the individual alarms. If you need more points, double up some of them so that one point covers two doors.

In case of an illegal entry, the program sounds a buzzer attatched to Port 1, bit 16. This buzzer turns on and off for 30 seconds prior to activation of the main siren. The siren stays on for two minutes then shuts off. After a 30-second wait, the program checks its status again and recycles if an alarm condition still exists.

There are also provisions in the program to make sure the relays shut off when they should. If the relay isn't off or fails to turn off the first time, there is one last attempt to shut it off. The system turns on the buzzer and prints an error message to attract your attention.

This is usually the result of a hardware problem. The relay may actually shut off but the indication read back is in error. An asterisk blinks once a second in the upper right-hand corner indicating the system is active.

I designed this system to be simple to use, fairlyburglar proof, and adaptable to your needs. It is not a guarantee of protection but only part of a home protection plan.

Sprinkler System Program

In Program Listing 3, I designate four sprinkler zones. It starts with a menu offering four options: Auto, Manual, Go, and Exit. This program is self-explanatory, but I'll briefly describe each mode.

The Auto mode patches the scan section of this program into the clock program. Once this is done, the program checks the time of day (am/pm), day code, and time to start. You can change this to suit your needs. When this mode starts, each zone runs for 15 minutes for a total of two cycles. The Go mode is the same as the Auto mode except that it starts on your request.

The Manual mode lets you turn on any zone for up to a maximum of 39 minutes. The program checks for illegal

Continued on p. 158

Listing 1 continued

```
ØØ43Ø          OUT     (ØECH),A       ;ENABLE MAIN BUS
ØØ44Ø          LD      A,144          ;SET 8255 FOR CONTROL
ØØ45Ø          OUT     (SETUP),A      ;CONFIGURE 8255
ØØ46Ø          LD      A,1            ;SET UP FOR READ
ØØ47Ø          OUT     (OUTC),A       ;PORT C FOR READ
ØØ48Ø          LD      C,OUTB         ;C IS USED FOR OUT INSTR.
ØØ49Ø          LD      B,12           ;USED TO ADDRESS COUNTERS
ØØ5ØØ          LD      HL,BUFFER      ;START OF TIME STORAGE
ØØ51Ø          CALL    GET            ;GET Y1Ø
ØØ52Ø          CALL    GET            ;GET Y1
ØØ53Ø          LD      (HL),'/'       ;PUT / IN WORD
ØØ54Ø          INC     HL             ;INC TO NEXT LOC IN BUFF
ØØ55Ø          CALL    GET            ;GET M1Ø
ØØ56Ø          CALL    GET            ;GET M1
ØØ57Ø          LD      (HL),'/'       ;PUT IN SEPERATOR
ØØ58Ø          INC     HL             ;INC TO NEXT POSITION
ØØ59Ø          CALL    GET            ;GET D1Ø
ØØ6ØØ          BIT     2,A            ;IS LEAP BIT SET?
ØØ61Ø          JR      Z,GETD1        ;SKIP OF NOT SET
ØØ62Ø          AND     33H            ;STRIP OFF LEAP BIT
ØØ63Ø          DEC     HL             ;BACK UP 1 POSITION
ØØ64Ø          LD      (HL),A         ;PUT CORRECT VALUE IN
ØØ65Ø          INC     HL             ;INC TO NEXT POSITION
ØØ66Ø GETD1    CALL    GET            ;GET D1
ØØ67Ø          CALL    FILL           ;INSERT 2 SPACES
ØØ68Ø          CALL    GET            ;GET DAY DIGIT (Ø-6)
ØØ69Ø          AND     Ø7H            ;MASK FOR DAY CODE
ØØ7ØØ          LD      (DIGIT),A      ;SAVE FOR SPINKLER PROG
ØØ71Ø          DEC     HL             ;RE-ALIGN BUFF PTR
ØØ72Ø          LD      DE,SU          ;GET ADDRESS OF TABLE
ØØ73Ø          RLA                    ;SHIFT FOR 2 WORD OFFSET
ØØ74Ø          ADD     A,E            ;ADD DAY INDEX TO A
ØØ75Ø          LD      E,A            ;PUT INDEX TO A REG
ØØ76Ø          PUSH    BC             ;SAVE PORT & ADRSS CONTRL
ØØ77Ø          LD      A,(DE)         ;GET PTR TO DAY CODE
ØØ78Ø          LD      (TEMP1),A      ;SAVE LSB OF DAY CODE
ØØ79Ø          INC     DE             ;INDEX TO GET NEXT PTR
ØØ8ØØ          LD      A,(DE)         ;GET NEXT PTR TO DAY CODE
ØØ81Ø          LD      (TEMP2),A      ;SAVE MSB OF DAY CODE
ØØ82Ø          LD      DE,(TEMP1)     ;GET ADDRESS OF DAY
ØØ83Ø          EX      DE,HL          ;SWAP PTRS FOR LDIR MOVE
ØØ84Ø          LD      BC,3           ;SET CTR TO MOVE 3 BYTES
ØØ85Ø          LDIR                   ;MOVE IT
ØØ86Ø          EX      DE,HL          ;RETORE PTRS.
ØØ87Ø          POP     BC             ;RESTORE PORT & ADDRESS'S
ØØ88Ø          CALL    FILL           ;FILL WITH SPACES
ØØ89Ø          CALL    GET            ;GET H1Ø
ØØ9ØØ          DEC     HL             ;SET BACK BUFF PTR
ØØ91Ø          PUSH    AF             ;SAVE A REG FOR AM CHECK
ØØ92Ø          AND     33H            ;MASK FOR H1Ø VALUE
ØØ93Ø          LD      (HL),A         ;PUT H1Ø VALUE TO BUFFER
ØØ94Ø          INC     HL             ;NEXT BUFFER LOCATION
ØØ95Ø          CALL    GET            ;GET H1
ØØ96Ø          LD      (HL),':'       ;PUT IN COLON
ØØ97Ø          INC     HL             ;NEXT BUFFER LOCATION
ØØ98Ø          CALL    GET            ;GET M1Ø
ØØ99Ø          CALL    GET            ;GET M1
Ø1ØØØ          LD      (HL),':'       ;PUT IN COLON
Ø1Ø1Ø          INC     HL             ;NEXT BUFFER LOCATION
Ø1Ø2Ø          CALL    GET            ;GET S1Ø
Ø1Ø3Ø          CALL    GET            ;GET S1
Ø1Ø4Ø          CALL    FILL           ;FILL WITH SPACES
Ø1Ø5Ø          POP     AF             ;RESTOR AM/PM INFO
Ø1Ø6Ø          BIT     3,A            ;IS THIS 24 HOUR FORMAT?
Ø1Ø7Ø          JR      Z,AMPM         ;SKIP AM/PM DISPLAY SECT
Ø1Ø8Ø          CALL    FILL           ;BLANK WHERE AM WAS TO BE
Ø1Ø9Ø          JR      DISPLY         ;GO AND DISPLAY TIME
Ø11ØØ AMPM     BIT     2,A            ;IS PM BIT SET?
Ø111Ø          JR      NZ,PM          ;PM BIT SET
Ø112Ø AM       LD      (HL),'A'       ;AM CODE
Ø113Ø          JR      CONT
Ø114Ø PM       LD      (HL),'P'       ;PM CODE
Ø115Ø CONT     INC     HL             ;INC TO NEXT BUFFER PTR
Ø116Ø          LD      (HL),'M'       ;PUT IN THE M FOR AM/PM
Ø117Ø DISPLY   LD      HL,BUFFER      ;BEGIN OF TIME INFO
Ø118Ø          LD      DE,3C24H       ;DESTINATION TO SCREEN
Ø119Ø          LD      BC;27          ;NUMBER OF TIME WORDS
Ø12ØØ          LDIR                   ;MOVE THE TIME TO SCREEN
Ø121Ø NODISP   LD      A,15           ;FOR CLOCK ADDRESSES
Ø122Ø          OUT     (OUTB),A       ;SET ADDRESS LINES HIGH
Ø123Ø          LD      A,129          ;SET READ HIGH-ENABLE INT
Ø124Ø          OUT     (OUTC),A       ;TURN ON INTRUPTS VIA PC7
Ø125Ø          XOR     A              ;CLEAR A REG
Ø126Ø          OUT     (ØECH),A       ;TURN OFF TRS MAIN BUS
Ø127Ø WATER    NOP                    ;THIS IS WHERE
Ø128Ø WATER1   NOP                    ;  THE SPRINKLER SYSTEM
Ø129Ø WATER2   NOP                    ;     IS PATCHED IN.
Ø13ØØ BURG     NOP                    ;THIS IS WHERE
Ø131Ø BURG1    NOP                    ;  THE BURGLER ALARM
Ø132Ø BURG2    NOP                    ;     IS PATCHED IN.
Ø133Ø          POP     AF             ;RESTORE REGISTORS
Ø134Ø          POP     IY
Ø135Ø          POP     IX
Ø136Ø          POP     BC
Ø137Ø          POP     DE
Ø138Ø          POP     HL
Ø139Ø          EI                     ;ENABLE INTERRUPTS
Ø14ØØ          RET                    ;RETURN TO INTERRUPTED PT
Ø141Ø ;
Ø142Ø ;        INDEX TO DAY MESSAGE TABLE
Ø143Ø ;
Ø144Ø SU       DEFW    SUN
Ø145Ø MO       DEFW    MON
Ø146Ø TU       DEFW    TUE
Ø147Ø WE       DEFW    WED
```

Listing 1 continued

Listing 1 continued

```
TR      DEFW    THR
FR      DEFW    FRI
SA      DEFW    SAT
;
;
DIGIT   DEFB    00              ;USED FOR SPRINK DAY CODE
TEMP1   DEFB    0               ;LSB STORAGE FOR DAY
TEMP2   DEFB    0               ;MSB STORAGE FOR DAY
BUFFER  DEFS    28              ;TIME INFORMATION BUFFER
;
;       TABLE OF DAY MESSAGES
;
SUN     DEFM    'SUN'
MON     DEFM    'MON'
TUE     DEFM    'TUE'
WED     DEFM    'WED'
THR     DEFM    'THR'
FRI     DEFM    'FRI'
SAT     DEFM    'SAT'
;
;       SUBROUTINE TO READ THE CLOCK'S COUNTERS
;
GET     OUT     (C),B           ;SET UP ADDRSS FOR READ
        OUT     (C),B           ;DO AGAIN FOR TIME DELAY
        IN      A,(INPUTA)      ;READ TIME
        ADD     A,30H           ;OBTAIN ASCII VALUE
        LD      (HL),A          ;PUT VALUE TO BUFFER
        INC     HL              ;INC TO NEXT BUFFER PTR
        DEC     B               ;DEC B FOR NEXT COUNTER
        RET                     ;FINISHED!
;
;       ROUTINE TO PUT SPACES BETWEEN TIME INFO
;
FILL    LD      (HL),20H        ;ASCII SPACE CODE
        INC     HL              ;INC NEXT SPACE
        LD      (HL),20H        ;PUT IN NEXT SPACE
        INC     HL              ;INC BUFFER PTR.
        RET                     ;DONE
;
        END     BEGIN
```

Program Listing 2. Burglar Alarm.

```
;       BURGLER ALARM PROGRAM
;       WRITTEN BY DAVE ENGELHARDT      10/82
;
;       MAY BE EXECUTED WITH A /30992....OR
;         COMMAND 'ALARM', WITH USE OF -CMDTBL- PROGRAM.
;
        ORG     7910H
BASIC   EQU     1A19H           ;JUMP TO BASIC
BURG    EQU     7F99H           ;EXTERNAL IN CLK
BURG1   EQU     7F9AH           ;EXTERNAL IN CLK
BURG2   EQU     7F9BH           ;EXTERNAL IN CLK
CLEAR   EQU     1C9H            ;CLEAR SCREEN
DORDLY  EQU     120             ;2 MIN TIME DELAY
ENTDLY  EQU     31              ;31 SEC ENT DLY (ODD VAL)
SIREN   EQU     120             ;2 MIN SIREN ON TIME
TOCLK   EQU     7F9CH           ;EXTERNAL JUMP TO CLK
;**************************
;         START OF PROGRAM
;**************************
ALARM   CALL    CLEAR           ;CLS
        LD      HL,MESS1
        CALL    21BH            ;PRINT INTRO MESSAGE
        LD      A,(ALSTAT)      ;TEST TO SEE IF ALARM
        CP      0               ;   ACTIVATED OR DEACT
        JR      NZ,ALMACT       ;GO IF ACTIVATED
        LD      HL,MESS2
        CALL    21BH            ;PRINT DEACT MESSAGE
        JR      AA              ;SKIP ACTIVATED MESSAGE
ALMACT  LD      HL,MESS3
        CALL    21BH            ;PRINT ACTIVATED MESSAGE
AA      LD      HL,MESS4
        CALL    21BH            ;PRINT CODE MESS
        LD      B,4             ;3 TRYS FOR CODE-LOOP
        CALL    CODE            ;GO AND TEST FOR CODE
        CALL    BUSON           ;TURN ON TRS INTERNAL BUS
        IN      A,(1H)          ;INPUT RELAY ALARM PORT
        CPL                     ;COMPLIMENT VALUE
        AND     30H             ;MASK RELAY BITS 4 & 5
        CP      0               ;ARE THE ALARMS OFF?
        JR      Z,ALT           ;SKIP IF RELAYS ARE OFF
        OUT     (1H),A          ;TURN OFF RELAYS
        CALL    DLY             ;GIVE THEM TIME TO LATCH
        JR      Z,ALT           ;SKIP IF OFF FOR SURE
        LD      HL,ALMFLT
        CALL    21BH            ;PRINT RELAY FAULT MESS
        JR      OFFA            ;GO AND DEACT SYSTEM
ALT     LD      A,(ALSTAT)      ;FLAG USED TO PUT
        CPL                     ;  SYSTEM IN IT'S
        LD      (ALSTAT),A      ;    OPPOSITE STATE
        CP      0               ;IS IT SET?
        JR      NZ,ACTALM       ;GO TO ACT SYSTEM
        CALL    CLEAR           ;CLEAR SCREEN
        LD      HL,MESS2
        CALL    21BH            ;PRINT DEACTIVATE MESS
;**************************
;       DISABLE ALARM PROGRAM
;**************************
OFFA    XOR     A               ;CLR A
```

Listing 2 continued

My wonderful upgrade offer:
If you bought my accounting software a while back, it's good news.
If you didn't, it's a good reason to buy it now.

You're probably growing. My software keeps growing too.

I've tinkered with it right along, improving it constantly. My accounting systems now work for CP/M, TRSDOS and MS/DOS (the IBM PC). It's a natural outgrowth of my support. Talk to a few thousand users on the phone, and you get a few new ideas.

Early on, I resolved that none of my customers would suffer if they bought a system before I improved it. So I've made this offer ever since I started over four years ago:

No matter when you bought, I'll upgrade any of my accounting systems to its latest capability on the same machine for $25. If you've upgraded your machine, I'll give you a replacement system for either $25 or the difference between what you paid for your software originally and the price of the new software. If you've gone from a TRS-80 Model I to a Model II/16 or an IBM PC, say, you get a full credit for whatever you paid me for your Model I systems. Just send me your old disks and I'll send you the new ones.

It's this simple. If you ever buy any of my software, you'll never lose your investment.

I wish the whole world were that simple.

Taranto
& ASSOCIATES, INC.

✓70

Model I, Model III and Model 4 systems: Accounts Payable, Accounts Receivable, General Ledger, Inventory Control, Invoicing, Payroll.

Model II, Model II/12/16, CP/M and IBM PC systems: General Ledger, Accounts Payable/Purchase Order, Accounts Receivable (Open Item or Balance Forward), Payroll/Job Costing, Inventory Control.

Post Office Box 6216, 121 Paul Drive, San Rafael CA 94903. Outside California, toll free (800) 227-2868. In California, (415) 472-2670.

✓ See List of Advertisers on Page 291

Listing 2 continued

```
        DI                      ;DISABLE INTERRUPTS
        LD      (BURG),A        ;CLEAR OUT JUMP
        LD      (BURG1),A       ;  FROM CLK TO
        LD      (BURG2),A       ;    ALARM PROGRAM
        LD      (ALMFLG),A      ;CLEAR SYSTEM FLAGS

        LD      (BYPASS),A      ;CLEAR TIME DELAY FLAGS
        LD      (ALSTAT),A      ;CLEAR SWAP FLAG
        EI                      ;ENABLE INTERRUPTS
        JP      BASIC           ;JUMP TO BASIC
;*************************
;  ENABLE ALARM PROGRAM
;*************************
ACTALM  XOR     A               ;CLR A REG
        LD      (SHFCTR),A      ;CLR FOR ALARM MESSAGES
        LD      (MASK1),A       ;CLR PORT 2 MASK
        LD      (MASK2),A       ;CLR PORT 3 MASK
;*********************************************
; SCAN ALARMS FOR SELECTIVE DEACTIVATION
;*********************************************
UPDATE  CALL    SECT1           ;SET-UP FOR PORT 2 ALRMS
        CALL    CHECK           ;PRINT PORT 2 ALARMS
        CALL    SECT2           ;SET-UP FOR PORT 3 ALRMS
        CALL    CHECK           ;PRINT PORT 3 ALARMS
D1      CALL    49H             ;SCAN KEYBD
        CP      'R'             ;RE-SCAN & DISPLAY ALARMS
        JR      Z.ACTALM        ;GO BACK AND RE-SCAN
        CP      'C'             ;CONTINUE WITH PROGRAM
        JR      NZ.D1           ;GO BACK TO KEYBD SCAN
        LD      HL,MESS6        ;PRINT ALARMS DEACT MESS
        CALL    21BH            ;PRINT ABOVE DEACT MESS
        LD      A,5H            ;SET FOR TMLOOP
TMLOOP  PUSH    AF              ;SAVE A CTR FOR TIME DLY
        CALL    60H             ;CALL TIME DELAY ROUTINE
        POP     AF              ;GET A BACK AND
        DEC     A               ;  DEC LOOP CTR
        CP      0               ;IS A ZERO YET?
        JP      NZ.TMLOOP       ;GO BACK IF NOT DONE
        CALL    CLEAR           ;CLS
        LD      HL,MESS7        ;PRINT DO YOU WANT TIME
        CALL    21BH            ;  DELAY MESS?
QTD     CALL    49H             ;QUESTION FOR TIME DELAY
        CP      'Y'             ;  IF NO -- CONTINUE
        JR      Z.DLYARM        ;DO TIME DELAY
        CP      'N'             ;CONTINUE IF NO
        JR      NZ.QTD          ;RE-SCAN KEYBD
ARM     LD      HL,BYPASS       ;GET TIME DLY CONTRL BYTE
        SET     0,(HL)          ;SET TO BYPASS ENTR DELAY
        JR      ARMIT           ;SKIP TIME DELAY SETUP
DLYARM  LD      HL,BYPASS       ;GET TIME DLY CONTRL BYTE
        SET     1,(HL)          ;SET FOR ENTRANCE DELAY
        LD      HL,TIMER        ;GET EXIT DELAY VALUE
        LD      (HL),DORDLY     ;2 MIN DOOR EXIT DELAY
;                                 AND 30 SEC ENTR DELAY
;*****************************
; PATCH ALARM PROGRAM TO CLK
;*****************************
ARMIT   LD      A,0C3H          ;PATCH JUMP INSTR TO CLK
        DI                      ;DISABLE INTERRUPTS
        LD      (BURG),A        ;1ST PART OF JUMP INSTR.
        LD      HL,CONECT       ;GET JUMP ADDRESS IN ALRM
        LD      (BURG1),HL      ;LOAD TO CLK ALARM JUMP
        XOR     A               ;A R M  T H E
        LD      (ALMFLG),A      ;  B U R G L E R  S Y S
        CALL    BUSON           ;TURN ON INTERNAL BUS
        IN      A,(2H)          ;GET AND STORE
        LD      (MASK1),A       ;  PORT 2 MASK
        IN      A,(3H)          ;GET AND STORE
        LD      (MASK2),A       ;  PORT 3 MASK
        CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        CALL    CLEAR           ;CLR SCREEN
        LD      HL,MESS3        ;PRINT SYSTEM
        CALL    21BH            ;  ACTIVATED MESSAGE
        EI                      ;ENABLE INTERRUPTS
        JP      BASIC           ;JUMP TO BASIC
;***************************************
; SECTION SCANNED BY CLK ONCE A SECOND
;***************************************
CONECT  LD      HL,3C3FH        ;GET SCREEN LOC FOR '*'
        LD      A,(SWITCH)      ;GET SWAP CONTROL BYTE
        CPL                     ;COMPLIMENT VALUE
        LD      (SWITCH),A      ;STORE VALUE BACK
        CP      0               ;IS IT ZERO?
        JR      Z.ASTRIK        ;SKIP IF ZERO
        LD      (HL),' '        ;PUT BLANK TO SCREEN
        JR      ASTRIK+2        ;SKIP
ASTRIK  LD      (HL),'*'        ;PUT '*' TO SCREEN
        LD      HL,BYPASS       ;GET TIME DELAY BYTE
        BIT     1,(HL)          ;IS IT SET FOR DELAY?
        JP      NZ.TMEOUT       ;GO TO COUNTDWN ROUTINE
        LD      HL,ALMFLG       ;GET ALARM CONTROL WORD
        BIT     5,(HL)          ;SET FOR SIREN?
        JR      NZ.TMEOUT       ;DO 30 SEC OFF INTERVAL
        BIT     6,(HL)          ;IS MAIN SIREN ON?
        JR      NZ.CNTDWN       ;GO AND COUNT DWN ON TIME
        BIT     4,(HL)          ;SET TO ALTERNATE SONA
        JR      NZ.T30SEC       ;BEEP SONA FOR 30 SEC
        BIT     3,(HL)          ;SET TO BYPASS ALARM SCAN
        JR      NZ.SKIP         ;  UNTIL SIREN CYCLE DONE
        CALL    ACTSCN          ;SCAN PORT 2 FOR ALARMS
        CALL    CHECK           ;ARE THERE ANY ALARMS?
        CALL    SECT2           ;SCAN PORT 3 FOR ALARMS
        CALL    CHECK           ;ARE THERE ANY ALARMS?
        LD      HL,ALMFLG       ;GET ALARM CONTROL BYTE
```

Listing 2 continued

Listing 2 continued

```
        BIT     3,(HL)          ;WILL BE SET ON ALRM COND
        JP      Z,TOCLK         ;JUMP BACK TO CLK
SKIP    BIT     7,(HL)          ;IS ENTRANCE TIME DLY SET
        JR      NZ,T30SEC       ;DONT TURN ON SIREN YET
NODLAY  CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,16            ;VALUE FOR SIREN
        OUT     (1H),A          ;TURN ON SIREN
        CALL    BUSOFF          ;SHUT OFF INTERNAL BUS
        LD      HL,ALMFLG       ;GET ALARM CONTROL BYTE
        SET     6,(HL)          ;SET TO FLAG SIREN ON THE
        LD      HL,TIMER        ;GET TIME CONTROL WORD
        LD      (HL),SIREN      ;LOAD SIREN WAILING TIME
        JR      CNTDWN          ;GO AND DEC TIME UNTIL=0
T30SEC  LD      HL,TIMER        ;GET TIME CONTROL WORD
        XOR     A               ;CLR A FOR COMPARE
        DEC     (HL)            ;DEC ENTRANCE DLY 30 SEC
        CP      (HL)            ;TEST HL FOR ZERO
        JR      Z,NODLAY        ;TIME DONE-TURN ON SIREN
        LD      HL,ALMFLG       ;GET CONTROL BYTE
        SET     4,(HL)          ;SET TO TURN ON SONA
        BIT     1,(HL)          ;BIT USED TO SWITCH
        JR      NZ,SWAP         ;  SONA (ON-OFF) TO ALERT
        SET     1,(HL)          ;    USER TO SHUT SYSTEM
        JR      OUTSON          ;      OFF UPON ENTRY
SWAP    RES     1,(HL)          ;RESET ALT FLAG
OUTSON  CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,32            ;VALUE TO TURN ON SONA
        OUT     (1H),A          ;TURN ON OR OFF SONA
        CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        JP      TOCLK           ;JUMP TO CLK PROGRAM
CNTDWN  LD      HL,TIMER        ;COUNTDWN SECTION TIMER
        XOR     A               ;CLR A FOR COMPARE
        DEC     (HL)            ;DEC TIME DELAY
        CP      (HL)            ;TEST FOR ZERO
        JP      NZ,TOCLK        ;SKIP IF NOT ZERO TO CLK
        CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,16
        OUT     (1H),A          ;TURN OFF SIREN
        CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        LD      HL,ALMFLG       ;SET-UP FOR OFF INTERVAL
        SET     5,(HL)          ;SET FOR SIREN OFF TIME
        LD      HL,TIMER        ;GET TIMER CONTROL WORD
        LD      (HL),ENTDLY     ;31 SEC PAUSE BWTN CYCLE
TMEOUT  LD      HL,TIMER        ;PAUSE INTERVAL ROUTINE
        XOR     A               ;CLR A FOR ZERO COMPARE
        DEC     (HL)            ;DEC TIME
        CP      (HL)            ;TEST FOR ZERO
        JP      NZ,TOCLK        ;GO IF TIME LEFT
        XOR     A               ;CLR A
        LD      (ALMFLG),A      ;CLR CONTROL WORD
        LD      HL,BYPASS       ;  AND GO FOR NEW CYCLE
        RES     1,(HL)          ;DISABLE ENTRANCE DELAY
        JP      TOCLK           ;RETURN TO CLK PROGRAM
;**************************
;       ALARM SCAN SECTION
;**************************
SECT1   CALL    CLEAR           ;CLR SCREEN
        LD      HL,MESS5
        CALL    21BH            ;PRINT ALARM STATUS MESS
ACTSCN  LD      A,(MASK1)       ;GET MASK FOR ALARMS THAT
        LD      B,A             ;  HAVE BEEN DEACTIVATED
        CALL    BUSON           ;TURN ON INTERNAL BUS
        IN      A,(2H)          ;READ PORT 2 FOR ALARMS
        CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        LD      HL,MTBLE1       ;GET START OF PORT 2 MESS
        LD      (MTBLE),HL      ;  TBLE & SAVE LOCATION
        RET
SECT2   LD      A,(MASK2)       ;GET MASK FOR ALARMS THAT
        LD      B,A             ;  HAVE BEEN DEACTIVATED
        CALL    BUSON           ;TURN ON INTERNAL BUS
        IN      A,(3H)          ;READ PORT 3 FOR ALARMS
        CALL    BUSOFF          ;TURN OF INTERNAL BUS
        LD      HL,MTBLE2       ;GET START OF PORT 3 MESS
        LD      (MTBLE),HL      ;  TBLE & SAVE LOCATION
        RET
CHECK   XOR     B               ;MASK OUT DEACT. ALARMS
        JR      Z,RETURN        ;ZERO MEANS NO ALARMS
        LD      HL,ALMFLG       ;GET ALARM CONTROL WORD
        SET     3,(HL)          ;SET TO BYPASS SECT CALLS
        CP      1               ;ALARM AT FRONT DOOR?
        JR      Z,EE            ;ZERO = IS YES THERE IS
        CP      2               ;ALARM AT GARAGE-HSE ENT?
        JR      NZ,DD           ;ZERO = IS YES THERE IS
EE      LD      HL,BYPASS       ;GET TIME DLY CONTL BYTE
        BIT     0,(HL)          ;IS 30 SEC TIME DLY SET?
        JR      NZ,DD           ;SKIP IF YES
        PUSH    AF              ;SAVE ALARM BITS
        LD      HL,ALMFLG       ;GET ALARM CONTROL BYTE
        SET     7,(HL)          ;SET FOR ENTR DELY 30 SEC
        LD      A,ENTDLY        ;LOAD ENTRANCE DELAY TIME
        LD      (TIMER),A       ;  TO TIME CONTROL BYTE
        POP     AF              ;RESTORE ALARM BITS
DD      LD      B,A             ;SAVE BIT PATTERN
C1      LD      D,0             ;CLR D REGISTER
        BIT     0,B             ;IS BIT 0 SET FOR ALARM?
        JR      Z,SHIFT         ;SKIP IF NOT AND TEST AGN
        LD      HL,(MTBLE)      ;GET CURRENT TABLE ADDRES
        LD      A,(SHFCTR)      ;GET MESSAGE CTR OR INDEX
        LD      E,A             ;STORE INDEX INTO E
        ADD     HL,DE           ;GET APPROPRIATE MESS
        LD      E,(HL)          ;GET LSB LOCATION OF MESS
        INC     HL              ;INC PTR TO MSB LOCATION
        LD      D,(HL)          ;GET MSB LOCATION OF MESS
        EX      DE,HL           ;PUT MESS LOCATION TO HL
```

Listing 2 continued

Listing 2 continued

```
02680         CALL    21BH             ;PRINT ALARM MESSAGE
02690 SHIFT   LD      A,(SHFCTR)       ;GET MESSAGE CTR AND INC
02700         ADD     A,2              ;  CTR FOR NEXT MESS LOC
02710         CP      16               ;ALL 8 MESS BEEN PRNTED?
02720         JR      Z,PRETRN         ;RETURN IF YES
02730         LD      (SHFCTR),A       ;SAVE NEW INDEX #
02740         RR      B                ;SHIFT B RIGHT 1 BIT
02750         JR      C1               ;GO TEST NEW BIT FOR ALRM
02760 PRETRN  XOR     A                ;PORT HAS BEEN SCANNED
02770         LD      (SHFCTR),A       ;CLR FOR NEXT PORT CHECK
02780 RETURN  RET                      ;RETURN FROM CALL
02790 ;****************************************************
02800 ; SECRET CODE ROUTINE TO ACT. OR DEACT ALARM PROGRAM
02810 ;                      CODE = MOM
02820 ;****************************************************
02830 CODE    DEC     B                ;DEC CTR AND TEST FOR
02840         LD      A,B              ;  ONLY THREE TRIES TO
02850         CP      0                ;    GET CODE CORRECT
02860         JR      Z,BAIL           ;CODE ERROR...EXIT PROG
02870         CALL    49H              ;SCAN KEYBD FOR CODE
02880         CP      'M'              ;IS FIRST CHARCT. A 'M'
02890         JR      NZ,CODE          ;RE-SCAN KEYBD IF NOT
02900         CALL    49H              ;SCAN KEYBD
02910         CP      'O'              ;IS 2ND CHARCT. A 'O'
02920         JR      NZ,CODE          ;RE-SCAN KEYBD IF NOT
02930         CALL    49H              ;SCAN KEYBD
02940         CP      'M'              ;IS LAST CHARCT. A 'M'
02950         JR      NZ,CODE          ;DEC B REG AND BAIL OUT
02960         RET
02970 ;**********************************************
02980 ; ROUTINE TO PRINT CODE ERROR & JUMP TO BASIC
02990 ;**********************************************
03000 BAIL    CALL    CLEAR            ;CLR SCREEN
03010         LD      HL,ERROR
03020         CALL    21BH             ;PRINT CODE ERROR MESS
03030         JP      BASIC            ;JUMP TO BASIC
03040 ;**********************************
03050 ; ROUTINE TO TURN INTERNAL BUS ON
03060 ;**********************************
03070 BUSON   PUSH    AF               ;SAVE A REG AND FLAGS
03080         LD      A,16
03090         OUT     (0ECH),A         ;ENABLE INTERNAL BUS
03100         POP     AF               ;RESTORE A REG AND FLAGS
03110         RET
03120 ;**********************************
03130 ; ROUTINE TO TURN INTERNAL BUS OFF
03140 ;**********************************
03150 BUSOFF  PUSH    AF               ;SAVE A REG AND FLAGS
03160         XOR     A
03170         OUT     (0ECH),A         ;TURN OFF INTERNAL BUS
03180         POP     AF               ;RETORE A REG AND FLAGS
03190         RET
03200 ;*********************************************
03210 ;  TIME DELAY USED TO ALLOW RELAYS TIME TO SWITCH
03220 ;*********************************************
03230 DLY     LD      BC,4FFFH         ;VALUE USED FOR CNT DWN
03240         CALL    60H              ;CALL TIME DLY
03250         IN      A,(1H)           ;READ RELAYS
03260         CPL                      ;COMPLIMENT VALUE
03270         AND     30H              ;MASK FOR ALARM RELAYS
03280         CP      0                ;ARE THEY OFF?
03290         RET
03300 ;******************
03310 ;    VARIABLES
03320 ;******************
03330 ALMFLG  DEFB    0                ;ALARM SYSTEM CONTRL BYTE
03340 ALSTAT  DEFB    0                ;BURGLER ALARM STATUS
03350 BYPASS  DEFB    0                ;TIME DELAY CONTROL BYTE
03360 MASK1   DEFB    0                ;SAVES PORT 2 DEACT ALRMS
03370 MASK2   DEFB    0                ;SAVES PORT 3 DEACT ALRMS
03380 MTBLE   DEFW    0                ;TABLE INDEX PTR.
03390 SHFCTR  DEFB    0                ;USED FOR MESS TBLE INDEX
03400 SWITCH  DEFB    0                ;USED TO BLINK '*'
03410 TIMER   DEFB    0                ;TIME DELAY CONTROL WORD
03420 ;******************************
03430 ;     MAIN MESSAGE TABLE
03440 ;******************************
03450 MESS1   DEFM    'THIS IS THE BURGLER ALARM PROGRAM'
03460         DEFW    0D0AH
03470 MESS2   DEFM    'THE ALARM SYSTEM IS * DEACTIVATED *'
03480         DEFB    0DH
03490 MESS3   DEFM    'THE ALARM SYSTEM IS ** ACTIVATED **'
03500         DEFB    0DH
03510 MESS4   DEFM    'ENTER (CODE WORD) TO PUT SYSTEM IN '
03520         DEFM    'OPPOSITE STATE'
03530         DEFW    0A0AH
03540         DEFM    '          ** OFF-ON  OR  ON-OFF**'
03550         DEFB    0DH
03560 MESS5   DEFB    0AH
03570         DEFM    'THESE ITEMS ARE IN ALARM CONDITION '
03580         DEFM    'CLOSE NEEDED POINTS'
03590         DEFB    0AH
03600         DEFM    'R = TO RECHECK     C = CONTINUE PROGRAM'
03610         DEFB    0DH
03620 MESS6   DEFM    '**** TAKE NOTE --ITEMS ABOVE ARE DE-'
03630         DEFM    'ACTIVATED ****'
03640         DEFB    0DH
03650 MESS7   DEFB    0AH
03660         DEFM    'DO YOU NEED TIME DELAYS OF *2 MINUTES '
03670         DEFM    'LEAVING  AND  30 SECONDS    '
03680         DEFM    'ENTERING* ON THE ENTRANCE DOORS? Y/N'
03690         DEFB    0DH
03700 ERROR   DEFB    0AH
03710         DEFM    '--CODE WORD--ENTRY ERROR..PROGRAM '
03720         DEFM    'EXITED!'
03730         DEFB    0DH
03740 ALMFLT  DEFM    'RELAY FAULT---SIREN OR SONA ILLEGALY ON'
```

Listing 2 continued

Continued from p. 153

zone and time entries. When it completes this mode, the program patches the Auto mode back in for a continued time scan. The Exit command leaves the program and jumps back to Basic.

This program also checks for relay failures and uses Port 4, bit 2 to sound a buzzer if a failure occurs. Upon entry to the program, it indicates the status of the system and the zone that is currently running if applicable. It displays a message giving you the option to deactivate the system or exit back to Basic. There is a 1½-minute delay between each zone that allows time for water pressure to stabilize.

CMDTABL Program

Program Listing 4 lets you patch custom commands to the Basic command table. Presently, I include only two commands—Sprink and Alarm. When you enter either of these two commands under Basic, the specified program runs. If you add more commands to the table, insert them in order but keep them between the ENDTBL label. This label automatically computes the length of the command table for scanning purposes.

Execute this program with a /29024. The set-up section patches the main body of this program to the Basic command table. Upon execution, two more commands are added to the system. This program runs anywhere in memory by changing the ORG statement.

Conclusion

You should now have a Port I/O board with relays to control the real world, and hopefully an understanding of how to control the port board and its functions. You should have enough information to expand in regard to port I/O control. The sprinkler and burglar alarm systems provide a couple of good applications to real-world control.

The amount of control applications available are virtually unlimited. CMDTBL opens a door to custom users for designing your own Basic commands and applications. You should learn a lot about your Model III and computers in general by putting together this system, as interfacing and controlling real world applications involve many aspects of your micro.

For a 16K System

As shown, the burglar alarm and sprinkler system don't assemble on a 16K RAM system since they include so many comments and banners. I made

Listing 2 continued

```
03750           DEFB    0DH
03760 ;****************************************
03770 ;     PORT 2 ALARM MESSAGE TABLE
03780 ;****************************************
03790 MTBLE1  DEFW    A0                  ;GIVES LOC FOR PORT 2
03800           DEFW    A1                  ;  ALARM MESSAGES
03810           DEFW    A2
03820           DEFW    A3
03830           DEFW    A4
03840           DEFW    A5
03850           DEFW    A6
03860           DEFW    A7
03870 ;****************************************
03880 ;     PORT 3 ALARM MESSAGE TABLE
03890 ;****************************************
03900 MTBLE2  DEFW    B0                  ;GIVES LOC FOR PORT 3
03910           DEFW    B1                  ;  ALARM MESSAGES
03920           DEFW    B2
03930           DEFW    B3
03940           DEFW    B4
03950           DEFW    B5
03960           DEFW    B6
03970           DEFW    B7
03980 ;********************************
03990 ;     PORT 2 ALARM MESSAGES
04000 ;********************************
04010 A0      DEFM    'FRONT DOOR'            ;** NOTE **.. A0
04020           DEFB    0DH                     ;  AND A1 MUST BE
04030 A1      DEFM    'GARAGE DOOR'           ;     USED AS ENTR
04040           DEFB    0DH                     ;        DLY INPUTS
04050 A2      DEFM    'FAMILY ROOM'
04060           DEFB    0DH
04070 A3      DEFM    'SLIDING DOORS'
04080           DEFB    0DH
04090 A4      DEFM    'KITCHEN WINDOW'
04100           DEFB    0DH
04110 A5      DEFM    'DINNING OR LIVING ROOM WINDOW'
04120           DEFB    0DH
04130 A6      DEFM    'SEWING ROOM'
04140           DEFB    0DH
04150 A7      DEFM    'A BATHROOM WINDOW'
04160           DEFB    0DH
04170 B0      DEFM    'MASTER BEDROOM'
04180           DEFB    0DH
04190 B1      DEFM    'BEDROOM #1 WINDOW'
04200           DEFB    0DH
04210 B2      DEFM    'BEDROOM #2 WINDOW'
04220           DEFB    0DH
04230 B3      DEFM    'BASEMENT WINDOW'
04240           DEFB    0DH
04250 B4      DEFM    'CAR #1 GARAGE DOOR'    ;NOTE: I HAVE TWO
04260           DEFB    0DH
04270 B5      DEFM    'CAR #2 GARAGE DOOR'
04280           DEFB    0DH
04290 B6      DEFM    'WATER LEAK IN BASEMENT';SENSE LEAKS
04300           DEFB    0DH
04310 B7      DEFM    'PANIC BUTTON'          ;FOR EMERGENCY
04320           DEFB    0DH
04330           END     ALARM
```

the listings easy to follow so you can assemble them on a 16K RAM system without the comments and banners. ■

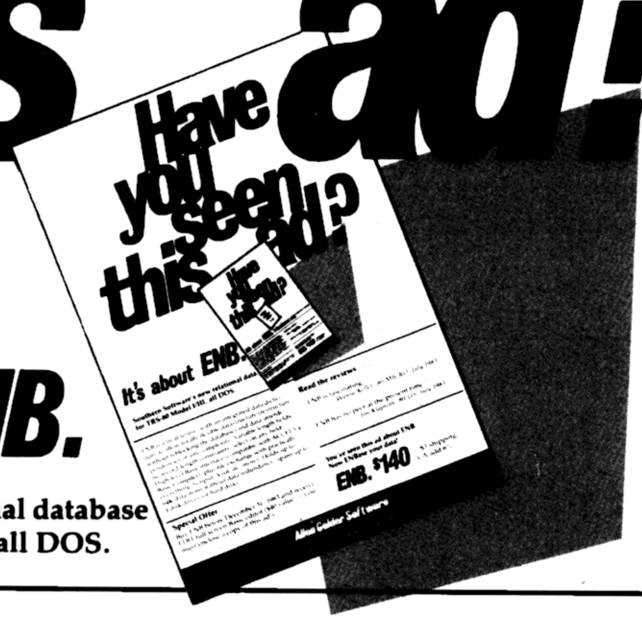
Contact David L. Engelhardt at 10221 W. 101 St. Place, Broomfield, CO 80020.

Program Listing 3. Sprinkler System.

```
;   SPRINKLER SYSTEM PROGRAM FOR A 4 ZONE SYSTEM
;   WRITTEN BY DAVE ENGELHARDT    10/82
;
;   EXECUTE WITH A /29152 ....OR USE
;     COMMAND 'SPRINK' FROM -CMDTBL- PROGRAM.
;
        ORG     71E0H
AMPM    EQU     7FD0H           ;WORD FOR AM/PM CODE
BASIC   EQU     1A19H           ;JUMP TO BASIC
CLEAR   EQU     1C9H            ;CLEAR SCREEN
DIGIT   EQU     7FB4H           ;DAY CODE FROM CLK PROGRM
DLYTME  EQU     90              ;TIME BETWEEN ZONES
HOURS   EQU     7FC6H           ;HOURS OBTAINED FROM CLK
MINUTS  EQU     7FC9H           ;MIN OBTAINED FROM CLK
RUNTME  EQU     900             ;THE TIME ZONES ARE ON
WATER   EQU     7F96H           ;PATCH SPRINK SYS TO CLK
WATER1  EQU     7F97H           ;PATCH TO CLK
WATER2  EQU     7F98H           ;PATCH TO CLK
TOCLK   EQU     7F99H           ;JUMP BACK TO CLOCK PROG
;************************
;   PROGRAM STARTS HERE
;************************
SPRINK  CALL    CLEAR           ;CLEAR SCREEN
        LD      A,(SPRFLG)      ;GET CONTROL BYTE
        CP      0               ;IS SYSTEM ENABLED
        JR      Z,JJ            ;GO TO SYSTEM STARTUP
;*****************************
; CHECK IF SYSTEM IS RUNNING
;*****************************
AUTCHK  BIT     7,A             ;IS RUNNING ACT BIT SET?
        JR      Z,MANCHK        ;SKIP TO CHECK MAN OPER
        LD      HL,MESS0
        CALL    21BH            ;PRINT AUTO ACT MESSAGE
        JP      ADYACT          ;JUMP TO STATUS SECTION
;****************************
; CHECK IF SYSTEM IS MANUEL
;****************************
MANCHK  BIT     1,A             ;IS MANUEL MODE BIT SET?
        JR      Z,AUTOMD        ;SKIP TO AUTO MODE SECT
        LD      HL,MESS4
        CALL    21BH            ;PRINT MAN MODE ACT MESS
        LD      HL,MESS14
        CALL    21BH            ;PRINT ZONE RUNNING
        LD      HL,MESS9A
        CALL    21BH            ;PRINT STOP SYSTEM MESS
        JP      PP              ;JUMP TO SCAN KEYBD
;********************
;  AUTO MODE SECTION
;********************
AUTOMD  LD      HL,MESS15
        CALL    21BH            ;PRINT AUTO MODE ACT MESS
        LD      HL,MESS9A
        CALL    21BH            ;PRINT STOP SYSTEM MESS
        JP      PP              ;JUMP TO SCAN KEYBD
;*********************************
;  SYSTEM INITIALIZATION SECTION
;*********************************
JJ      CALL    BUSON           ;TURN ON INTERNAL BUS
        IN      A,(1H)          ;READ SPRINKLER RELAYS
        CPL                     ;COMPLIMENT VALUE
        AND     0FH             ;MASK FOR SPRINK RELAYS
        CP      0               ;ARE THEY OFF?
        JR      Z,TT            ;SKIP IF YES
        OUT     (1H),A          ;TURN OFF RELAYS
        CALL    DLY             ;ALLOW RELAYS TO LATCH
        JR      Z,TT            ;SKIP IF RELAYS ARE OFF
        LD      HL,MESS13
        CALL    21BH            ;PRINT RELY STILL ON ERR
        JP      BASIC           ;JUMP TO BASIC
TT      CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        LD      A,0DH
        CALL    33H             ;DO CARRIAGE RETURN
        LD      HL,MESS1
        CALL    21BH            ;PRINT MENU
AA      CALL    49H             ;SCAN KEYBD FOR SELECTION
        CP      'M'
        JR      Z,MAN           ;GO TO MANUEL MODE
        CP      'A'
        JR      Z,AUTO          ;GO TO AUTO MODE
        CP      'G'
        JR      Z,KICK          ;GO START AUTO ON DEMAND
        CP      'E'
        JP      BASIC           ;JUMP TO BASIC
        JR      AA              ;RE-SCAN KEYBD
;******************************
;  SET SYSTEM UP FOR AUTO MODE
;******************************
AUTO    CALL    AUPATC          ;PATCH SYS FOR AUTO MODE
RR      CALL    SOUND           ;BEEP SONA TO ACKNOW ACT
        CALL    CLEAR           ;CLEAR SCREEN
        LD      HL,MESS0
        CALL    21BH            ;PRINT AUTO MODE ACTIVAT
        JP      BASIC           ;JUMP BACK TO BASIC
;*********************************
;  ROUTINE TO START AUTO ON DEMAND
;*********************************
KICK    CALL    AUPATC          ;PATCH AUTO MODE TO CLK
        LD      HL,SPRFLG       ;GET CONTROL WORD
        SET     7,(HL)          ;SET SYSTEM RUNNING BIT
        JR      RR              ;MAKE SOUND & PRINT MESS
;*******************************
;  ACTIVATE MANUEL MODE SECTION
;*******************************
MAN     LD      HL,SPRFLG       ;GET CONTROL WORD
        SET     1,(HL)          ;SET MANUEL MODE ACT BIT
        CALL    CLEAR           ;CLEAR SCREEN
        LD      HL,MESS2
        CALL    21BH            ;PRINT MANUEL MODE
CC      LD      HL,MESS3
        CALL    21BH            ;PRINT ZONE MENU
BB      CALL    49H             ;SCAN KEYBD FOR ZONE #
        CALL    33H             ;PRINT ZONE # TO SCREEN
        LD      (MZONE),A       ;STORE VALUE TO MESS 14
        SUB     30H             ;STRIP OFF ASCII VALUE
        CP      0               ;CORRECT VALUE?
        JR      Z,ZINERR        ;SKIP TO ILLEGAL ZONE #
        LD      (ZHOLD),A       ;SAVE ZONE #
        SUB     5               ;IS ZONE # LESS THEN 5
        JP      M,DD            ;IF NOT...SKIP TO DD
;**************************
;  ZONE INPUT ERROR SECTION
;**************************
ZINERR  CALL    CLEAR           ;CLEAR SCREEN
        LD      HL,MESS6
        CALL    21BH            ;PRINT ZONE ERROR MESS
        JP      CC              ;GO AND RE-ENTER ZONE #
;*****************************************************
;IF ENTERED ZONE VALUE IS OK--ENTER MINUTES INTO BUFFER
;*****************************************************
DD      LD      HL,MESS5        ;GET READY TO PRINT MESS
        LD      A,0DH
        CALL    33H             ;DO CARRIAGE CONTROL
        CALL    33H             ;DO CARRIAGE CONTROL
        CALL    21BH            ;PRINT ZONE TIME MESS
        CALL    49H             ;SCAN KEYBD FOR 1ST VALUE
        CALL    33H             ;PRINT VALUE TO SCREEN
        SUB     30H             ;STRIP ASCII VALUE
        LD      (TENS),A        ;SAVE TENS OF MINUTES
        SUB     4               ;MUST BE LESS THAN 40 MIN
        JP      M,FF            ;CONTINUE IF LESS THEN 40
        CALL    CLEAR           ;CLEAR SCREEN
        LD      HL,MESS8
        CALL    21BH            ;PRINT TIME ERROR MESS
        JR      DD              ;GO AND REDUE TIME ENTRY
FF      CALL    49H             ;SCAN KEYBD FOR TIME
```

Listing 3 continued on p. 164

Have you seen this ad?



It's about ENB.

Southern Software's new relational database manager for TRS-80 Model I/III, all DOS.

ENB is a treat to use, with an integrated data dictionary to allow totally flexible datastructure (restructure without reblocking the database) and data-interdependencies of any complexity. Variable length fields, no record-length constraints, select on any field. High-level Basic interface (compatible with ACCEL3/4 Basic Compiler), plus file exchange with practically everything (Scripsit, VisiCalc, more). Holds up to 64K data items without data redundancy, spans up to 4 disk drives (or hard disk).

Special Offer

Buy ENB before December 31, 1983 and receive EDIT full-screen Basic editor ($40 value — you must enclose a copy of this ad!).

Scripsit ™ Radio Shack, VisiCalc ™ VisiCorp,

Read the reviews . . .

"ENB is fascinating."
— *Wynne Keller, 80-MICRO, July 1983.*

"ENB has no peer at the present time."
— *Jim Klaproth, 80-US, July 1983.*

**You've seen this ad about ENB.
Now ENBase your data!**

VISA master charge

ENB. $140 + $3 shipping
CA add 6%

Allen Gelder Software

Box 11721 San Francisco, CA 94101 (415) 681-9371

✓136

✓ See List of Advertisers on Page 291

Listing 3 continued

```
         CALL    33H            ;PRINT VALUE TO SCREEN
         SUB     30H            ;STRIP ASCII VALUE
         LD      (UNITS),A      ;SAVE UNITS OF MINUTES
         LD      HL,MESS7       ;READY MESS FOR PRINT
         LD      A,0DH
         CALL    33H            ;DO CARRIAGE CONTROL
         CALL    33H            ;DO CARRIAGE CONTROL
         CALL    21BH           ;PRINT ARE YOU SATISFIED
         CALL    49H            ;SCAN KEYBD FOR ANSWER
         CP      'C'            ;C = TO CONTINUE PROGRAM
         JR      Z,EE           ;SKIP IF C
         CALL    CLEAR          ;CLEAR SCREEN
         JR      CC             ;GO & REDUE ZONE # & TIME
;**************************************************
;  ROUTINE TO CHANGE KYBD ENTRY TO HEX EQUIVALENT
;**************************************************
EE       LD      A,(TENS)       ;GET TENS OF MINUTES
         SLA     A              ;SHIFT VALUE LEFT
         PUSH    AF             ;SAVE CONTENTS
         SLA     A              ;SHIFT LEFT AGAIN
         SLA     A              ;SHIFT LEFT
         POP     BC             ;TRANSFER AF TO BC
         ADD     A,B            ;ADD VALUES TOGETHER
         LD      B,A            ;STORE VALUE TO A REG
         LD      A,(UNITS)      ;GET UNITS OF TIME
         ADD     A,B            ;ADD UNITS TO TENS/MIN
;**************************************************
;  ROUTINE TO CONVERT HEX ON-TIME INTO SECONDS
;**************************************************
         LD      (UNITS),A      ;STORE HEX TIME VALUE
         LD      L,A            ;PUT HEX VALUE TO L REG
         XOR     A              ;CLEAR A REG
         LD      H,A            ;CLEAR H REG
         SLA     L              ;SHIFT LEFT
         SLA     L              ;SHIFT LEFT
         PUSH    HL             ;SAVE SHIFTED VALUE
         SLA     L              ;SHIFT LEFT
         SLA     L              ;SHIFT LEFT
         CALL    HSHIFT         ;SHIFT H ACCORDING TO L
         SLA     H              ;SHIFT LEFT
         SLA     L              ;SHIFT LEFT
         CALL    HSHIFT         ;SHIFT H ACCORDING TO L
         SLA     H              ;SHIFT LEFT
         SLA     L              ;SHIFT LEFT
         CALL    HSHIFT         ;SHIFT H ACCORDING TO L
         POP     BC             ;PUT HL VALUE TO BC REG
         XOR     A              ;RESET CARRY FLAG
         SBC     HL,BC          ;CALCULATE FINAL VALUE
         LD      (TENS),HL      ;SAVE TIME IN SECONDS
;****************************************
;  ROUTINE TO PATCH MANUAL MODE TO CLOCK
;****************************************
SETUP    CALL    PORTKY         ;GET MANUEL ACT BIT
         LD      (MNPORT),A     ;SAVE VALUE FOR RELAY
         DI                     ;DISABLE INTERRUPTS
         LD      A,0C3H         ;PATCH SPRINKLER
         LD      (WATER),A      ;  SYSTEM TO THE
         LD      HL,MANRT       ;     CLOCK
         LD      (WATER1),HL    ;       PROGRAM
         EI                     ;ENABLE INTERRUPTS
         CALL    SOUND          ;BEEP SONA
         CALL    CLEAR          ;CLEAR SCREEN
         LD      HL,MESS14
         CALL    21BH           ;PRINT ACTIVE ZONE MESS
         JP      BASIC          ;JUMP BACK TO BASIC
;***************************
;  ROUTINE TO SHIFT H WITH L
;***************************
HSHIFT   JR      C,ADDBIT       ;CARRY BIT SET VIA SHIFT?
         JR      NOTSET         ;SKIP IF NOT SET
ADDBIT   SET     0,H            ;SET BIT 0
NOTSET   XOR     A              ;CLEAR CARRY BIT
         RET
;****************************************
;  ROUTINE TO INDICATE ZONE NOW RUNNING
;****************************************
ADVACT   LD      HL,MESS9       ;READY MESS TO PRINT
         LD      A,(ZHOLD)      ;GET RUNNING ZONE
         ADD     A,30H          ;OBTAIN ASCII VALUE
         LD      (ZNUM),A       ;STORE VALUE TO MESS
         CALL    21BH           ;PRINT ZONE RUNNING MESS
PP       CALL    49H            ;SCAN KEYBD
         CP      'N'            ;N = LEAVE PROG AS IS
         JP      Z,BASIC        ;BACK TO BASIC
         CP      'Y'            ;Y = DEACTIVATE SYSTEM
         JR      NZ,PP          ;RE-SCAN KEYBD
;***************************
;  ROUTINE TO SHUT OFF ZONE
;***************************
         CALL    BUSON          ;TURN ON INTERNAL BUS
         IN      A,(1H)         ;READ RELAY STATUS
         CPL                    ;COMPLIMENT VALUE
         AND     0FH            ;MASK FOR SPRINKLER RELYS
         CP      0              ;ARE THE RELAYS OFF?
         JP      Z,TOTAL        ;SKIP IF YES
         OUT     (1H),A         ;TURN RELAY OFF
         CALL    DLY            ;GIVE RELAY TIME TO LATCH
         JR      NZ,ERROR       ;GO TO ERROR IF STILL ON
;******************************
;  DEACTIVATE SPRINKLER SYSTEM
;******************************
TOTAL    CALL    CLEAR          ;CLEAR SCREEN
         LD      HL,MESS12
         CALL    21BH           ;PRINT DEACTIVATED MESS
         CALL    UNPATC         ;UNPATCH SPRINK FROM CLK
         CALL    SOUND          ;BEEP SONA
         JP      BASIC          ;JUMP TO BASIC
;*****************************************************
; ERROR ROUTINE--ZONE ILLEGALLY ON OR FAILED TO TURN OFF
;*****************************************************
ERROR    PUSH    AF             ;SAVE CONTENTS
         LD      A,(ZHOLD)      ;GET ZONE #
         ADD     A,30H          ;OBTAIN ASCII VALUE
         LD      (EZONE),A      ;  & PUT IN ERROR MESSAGE
         LD      HL,MESS10
         CALL    21BH           ;PRINT RELAY FAULT MESS
         CALL    BUSON          ;ENABLE INTERNAL BUS
         POP     AF             ;SAVE CONTENTS
         OUT     (1H),A         ;TRY 1 LAST SHUT-OFF
         LD      A,2
         OUT     (4H),A         ;TURN ON SONA SOLID
         CALL    UNPATC         ;UNPATCH SPRINK FROM CLK
         JP      BASIC          ;JUMP TO BASIC
;****************************************
;  UNPATCH SPRINKLER SYSTEM FROM CLOCK
;****************************************
UNPATC   XOR     A              ;CLEAR A REG
         DI                     ;DISABLE INTERRUPTS
         LD      (SPRFLG),A     ;CLEAR CONTROL BYTE
         LD      (WATER),A      ;UNPATCH
         LD      (WATER1),A     ;  FROM
         LD      (WATER2),A     ;    CLOCK
         EI                     ;ENABLE INTERRUPTS
         RET
;**********************
;  AUTO PATCH ROUTINE
;**********************
AUPATC   DI                     ;DISABLE INTERRUPTS
         XOR     A              ;CLEAR A REG
         LD      (ZHOLD),A      ;RESET ASCII ZONE VALUE
         LD      A,1
         LD      (SPRFLG),A     ;SET SYSTEM ACTIVE
         LD      (CYCLE),A      ;SET CYCLE CTR
         LD      (ZNUM),A       ;RESET ASCII VALUE MESS9
         LD      A,2
         LD      (CYK),A        ;SET CTR FOR TWO CYCLES
         LD      A,'1'
         LD      (ZCYK),A       ;SET CYCLE VALUE MESS9
```

Listing 3 continued

THE REST OF 80

IT HAD TO GO SOMEWHERE

There's no reason to deprive you of solid microcomputing information just because it wouldn't fit between the covers of *80 Micro*. But even *80*, as thick as it is, can hold just so much. Here's the answer—**The Rest of 80**—31 of the best tutorials and utilities, hand-picked from the overflowing files at *80 Micro*. These never-before-published articles for the Model I and Model III were just too good to let them get away.

SOMETHING FOR EVERYONE

Whatever your programming skills, **The Rest of 80** can help you learn more and save time and effort. Here are just a few of the chapters you'll find:

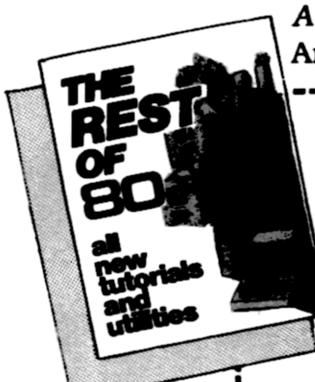
An Unlistable, Unbreakable Program
Adding Commands to BASIC
Programming in Tiny Pascal
Line Drawing
Automatic Master Disk Directory
Faster Loading for the Model I
ASCII Converter
A Better LDOS KSM

And more on BASIC, Pascal, and assembly language!

Every program is of the same high quality you wait for every month in *80 Micro*. Now try **The Rest of 80**—a welcome addition to your computer library.

ISBN 0-88006-062-X, softcover with spiral binding, 7×9, approx. 300 pp. BK7392 $9.97

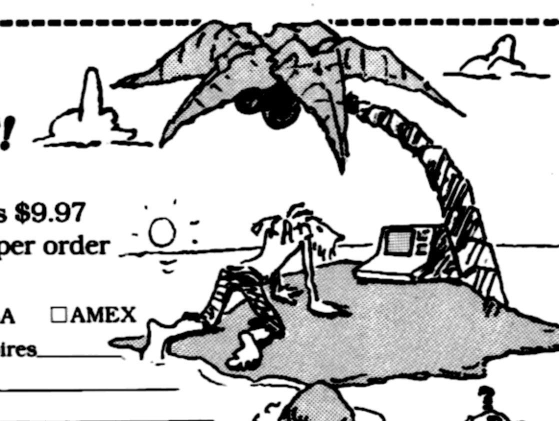
Call **TOLL-FREE 1-800-258-5473** for credit card orders. Or mail your order with payment or complete credit card information. Include $1.50 per order for shipping and handling. Send to: Wayne Green Inc., Attn: Book Sales, Peterborough, NH 03458.



A WAYNE GREEN PUBLICATION

WG BOOKS

I'D BE LOST WITHOUT IT!



Send me ____ copies of **THE REST OF 80**. Enclosed is $9.97 (BK7392) per copy plus $1.50 per order shipping and handling.

☐ MasterCard Bank #________ ☐VISA ☐AMEX

Card #______________________ Expires__________

Signature ______________________

Name ______________________

Address ______________________

City ______________________

State ______________ Zip______________

33DB8R

Photocopy of coupon is acceptable for ordering

Listing 3 continued

```
        LD      A,0C3H          ;PATCH SPRINKLER
        LD      (WATER),A       ;  SYSTEM TO
        LD      HL,AUTORT       ;    THE CLOCK
        LD      (WATER1),HL     ;      PROGRAM
        EI                      ;ENABLE INTERRUPTS
        RET
;*********************************************
;  MANUEL ROUTINE PATCHED TO CLOCK PROGRAM
;*********************************************
MANRT   LD      HL,SPRFLG       ;GET CONTROL WORD
        BIT     1,(HL)          ;MAKE SURE MANUEL MODE
        JR      Z.EXITMN        ;SKIP IF NOT SET
        BIT     2,(HL)          ;SET FOR ON-TIME COUNTDWN
        JR      NZ.SKIP         ;SKIP UNTIL TIME IS UP
        CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,(MNPORT)      ;LOAD ZONE ACTIVATE BIT
        OUT     (1H),A          ;TURN ON ZONE
        CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        LD      HL,SPRFLG       ;GET CONTROL WORD
        SET     2,(HL)          ;SET TO BYPASS ACT CODE
SKIP    LD      HL,(TENS)       ;LOAD SPRINKLER ZONE
        DEC     HL              ;  ON TIME & COUNT DWN
        LD      (TENS),HL       ;PUT TIME BACK
        LD      A,H             ;CHECK FOR
        OR      L               ;  TIME EQUAL
        CP      0               ;    TO ZERO
        JP      NZ.TOCLK        ;JUMP TO CLOCK
        CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,(MNPORT)      ;GET ZONE ACT BIT
        OUT     (1H),A          ;TURN OFF ZONE
        CALL    DLY             ;DELAY FOR RELAY TO LATCH
        JR      Z.EXITMN        ;SKIP IF ZONE TURNED OFF
        JP      ERROR           ;ERROR...ZONE STILL ON
EXITMN  CALL    AUPATC          ;PATCH AUTO & CLR FLAGS
        LD      HL,MESS11
        CALL    21BH            ;PRINT AUTO MODE MESS
        CALL    SOUND           ;ACKNOWLEGE MODE SET
        JP      TOCLK           ;JUMP TO CLK
;*********************************************
;  AUTO SPRINKLER MODE PATCHED TO CLK PROGRAM
;*********************************************
AUTORT  LD      HL,SPRFLG       ;GET CONTROL WORD
        BIT     7,(HL)          ;SET = SPRINKLER ACTIVE
        JR      NZ.AUT          ;BYPASS TIME CHECK
        LD      A,(AMPM)        ;READ A OR P FROM CLK
        CP      41H             ;IS IT AM?
        JP      NZ.TOCLK        ;LEAVE--NOT TIME TO START
        LD      A,(DIGIT)       ;GET DAY CODE FROM CLK
        CP      1               ;IS IT MONDAY?
        JR      Z.KK            ;SKIP AND START IF IT IS
        CP      3               ;IS IT WEDNESDAY?
        JR      Z.KK            ;SKIP AND START IF IT IS
        CP      5               ;IS IT FRIDAY?
        JR      Z.KK            ;SKIP AND START IF IT IS
        CP      8               ;THIS IS A SPARE DAY
        JP      NZ.TOCLK        ;LEAVE - NOT TIME TO STRT
KK      XOR     A               ;CLR CARRY FLG
        LD      HL,(HOURS)      ;GET ACTUAL TIME
        LD      BC,(HRS)        ;LOAD STRT TIME IN HRS
        SBC     HL,BC           ;ARE THEY EQUAL?
        JP      NZ.TOCLK        ;LEAVE AND KEEP TESTING
        XOR     A               ;CLR FLAGS
        LD      HL,(MINUTS)     ;GET ACTUAL MINUTES
        LD      BC,(MIN)        ;LOAD STRT TIME IN MIN
        SBC     HL,BC           ;ARE THEY EQUAL?
        JP      NZ.TOCLK        ;LEAVE AND KEEP TESTING
        LD      HL,SPRFLG       ;GET CONTROL WORD
        SET     7,(HL)          ;SET TO BYPASS TIME CHKS
AUT     LD      HL,SPRFLG       ;GET CONTROL WORD
        BIT     0,(HL)          ;TEST FOR AUTO ENABLED
        JP      Z.EXITMN        ;LEAVE IF AUTO NOT SET
        BIT     4,(HL)          ;SET FOR PAUSE DELAY
        JR      NZ.DELAY        ;SKIP FOR DELAY
        BIT     3,(HL)          ;ON TIME DELAY...ZONE ON
        JR      NZ.SKIP1        ;SKIP IF ZONE RUNNING
HH      CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,(CYCLE)       ;GET ZONE ACTIVATE BIT
        OUT     (1H),A          ;TURN ON ZONE
        CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        LD      HL,SPRFLG       ;GET CONTROL WORD
        SET     3,(HL)          ;SET TO SKIP HH SECTION
        LD      HL,ZHOLD        ;GET ZONE #
        INC     (HL)            ;INCREASE ZONE # BY 1
SKIP1   LD      HL,(TMEWRD)     ;GET ZONE ON TIME
        DEC     HL              ;DEC ON TIME ONCE/SECOND
        LD      (TMEWRD).HL     ;STORE TIME BACK
        LD      A,H             ;CHECK FOR
        OR      L               ;  TIME EQUAL
        CP      0               ;    TO ZERO
        JP      NZ.TOCLK        ;SKIP IF NOT
        LD      HL,RUNTME       ;GET ZONE ON TIME VALUE
        LD      (TMEWRD).HL     ;RESTORE ON TIME PERIOD
        CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,(CYCLE)       ;GET ZONE ACTIVATE BIT
        OUT     (1H),A          ;TURN OFF ZONE
        CALL    DLY             ;TIME FOR RELAY TO LATCH
        JP      NZ.ERROR        ;ERROR...RELAY STILL ON
        CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        LD      HL,SPRFLG       ;GET CONTROL WORD
        RES     3,(HL)          ;RESET TO ACT NEXT ZONE
        SET     4,(HL)          ;SET FOR PAUSE DELAY
DELAY   LD      HL,PAUSE        ;LOAD PAUSE TIME
        DEC     (HL)            ;DECREMENT PAUSE TIME
        XOR     A               ;CLR FOR COMPARE
        CP      (HL)            ;IS TIME = 0
        JP      NZ TOCLK        ;JUMP TO CLK IF NOT
        LD      HL,SPRFLG       ;GET CONTROL WORD
        RES     4,(HL)          ;RESET PAUSE DELAY
        LD      A,DLYTME        ;GET DELAY TIME VALUE
        LD      (PAUSE),A       ;RELOAD PAUSE TIME
        LD      A,(CYCLE)       ;GET ACTIVATE BIT
        SLA     A               ;SHFT LEFT = NEXT ACT BIT
        BIT     4,A             ;IF SET = AUTO MODE DONE
        JR      NZ.C1           ;CHECK FOR 2ND CYCLE
        LD      (CYCLE).A       ;PUT BACK SHIFTED BIT
        JP      TOCLK           ;GO FOR NEXT ZONE
;*********************************
;  ROUTINE TO CHECK FOR 2ND CYCLE
;*********************************
C1      LD      A,(CYK)         ;LOAD CYCLE COUNT
        DEC     A               ;DEC CYCLE
        CP      0               ;IS THIS 2ND PASS?
        JP      Z.EXITMN        ;SPRINKLER SYSTEM DONE
        LD      (CYK),A         ;LOAD DEC CYCLE # BACK
        LD      A,1             ;RESET SYSTEM TO ZONE #1
        LD      (CYCLE).A       ;  FOR SECOND PASS
        XOR     A
        LD      (ZHOLD).A       ;CLEAR ASCII ZONE #
        LD      A,'2'
        LD      (ZCYK).A        ;PUT IN MESS9 FOR PRINT
        JP      TOCLK           ;JUMP TO CLOCK
;**************************
;          VARIABLES
;**************************
HRS     DEFW    3530H           ;5 0   = TURN ON TIME
MIN     DEFW    3030H           ;0 0   = 05:00  AM
MNPORT  DEFB    0               ;Z CNTRL BIT FOR MAN
SPRFLG  DEFB    0               ;SPRINKLER CONTROL WORD
CYCLE   DEFB    1               ;SHIFTING ZONE ACT BIT
TMEWRD  DEFW    30              ;ZONE ON-TIME=15 MINUTES
PAUSE   DEFB    10              ;ZONE PAUSE TIME= 1.5 MIN
ZHOLD   DEFB    0               ;ZONE VALUE FOR ASCII PRNT
TENS    DEFB    0               ;USED FOR MAN TIME INPUT
UNITS   DEFB    0               ;" " " " " "  TIME INPUT
CYK     DEFB    2               ;CYLE CONTROL FOR SYSTEM
;*****************************************
;  THIS TABLE IS THE ACTIVATE BIT TABLE
;*****************************************
PORT    DEFB    0H              ;DUMMY VALUE
        DEFB    1H              ;ZONE 1 ACTIVATE BIT
        DEFB    2H              ;ZONE 2
```

Listing 3 continued

TCS Heart of TEXAS COMPUTER SYSTEMS

Toll Free 1-800-433-5184 Texas 1-817-274-5625

TCS Model IV, 64K, 2 Disks

Systems come with 180 Day TCS Warranty

$1499
With standard 40 track double density drives.
Over 340,000 bytes.
Enhanced Model IV Operating System

RS-232 FREE!

$1699
With 2 dual headed 40 track double density drives
Over 730,000 bytes
Enhanced Model IV Operating System

Fully assembled and tested systems that are software compatible and functionally identical to Radio Shack units sold at computer stores for $hundreds more.

- CONTROLLER BOARDS are high quality double sided epoxy boards with gold plated contacts.
- POWER SUPPLY is the finest switching type available.
- MOUNTING HARDWARE includes power and data cables.
- DISK DRIVES are Tandon, the same ones used by Radio Shack... 40 track, double density, with a 5 millisecond stepping rate.

TCS MODEL III DISK EXPANSION KITS

1 Controller, Power Supply, Mounting Hardware & Instructions	$249
2 Controller, Power Supply, Hardware & one 40 track Tandon Drive	$429
3 Controller, Power Supply, Hardware, two 40 track Tandon Drives	$598
3a Kit 3 but with two 80 track drives (dual sided 40s)	$791
3b Kit three but with two 160 track drives (dual sided 80s)	$989

TCS MODEL IV DISK EXPANSION KITS

11 Controller, Power Supply, Mounting Hardware, one 40 track Tandon Drive	$479
12 Controller, Power Supply, Mounting Hardware, two 40 Track Tandon Drives	$649
12A Kit 12 but with two 80 Track Tandon Drives	$629
12B Kit 12 but with two 160 Track Tandon Drives	$849

TRS-80

TRS-80 equipment below has original 90 day Manufacturer's Limited Warranty

MODEL 12 and MODEL 16

MODEL 12, 1 drive	$CALL
MODEL 12, 2 drives	$CALL
TCS MODEL 12 Version, 2 Tandon drives (like the original) TCS Warranty	$2995

MODEL 16B...Support up to 6 users. Run your whole office with hard disk capabilities for about $1000 per user!

MODEL 16B, 1 drive	$CALL
MODEL 16B, 2 drives	$CALL

Model 12 and Model 16 Accessories

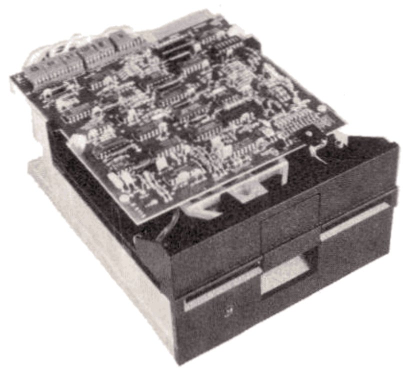
128K memory board (256K Max.)	$629
128K extra memory chips (RS)	$269
128K extra memory chips (TCS)	$189
Xenix Microsoft Multi-user Basic	$269
Xenix Accounting Software	$CALL
Xenix Multiplan Spread Sheet Software	$263
MII/12 to M16 multi-user upgrade kit	$1339
DT/1 Video Terminal	$629

MODEL IV

MODEL IV, 16K Cassette	$825
MODEL IV, 64K, 2 drives, RS-232	$CALL

Model III Color Computer

All Radio Shack equipment is shipped from our store in Brady, Texas



Tandon

TCS DRIVE CABINET is industrial grade heavy guage metal, safety fused, and comes with gold plated external connector with extender cable.

1 DRIVE in Cabinet

40 track single sided	$199
80 track (dual sided 40 track)	$299
160 track (dual sided 80 track	$399

2 DRIVE Double Cabinet

40 track single sided	$399
80 track (dual sided 40 tracks)	$599
160 track (dual sided 80 tracks)	$799

1 DRIVE Double Cabinet

40 track single sided	$269
80 track (dual sided 40 track)	$369
160 track (dual sided 80 track)	$449

Drives in cabinets come assembled and tested with power supply. Order cable separately.

BARE DRIVES ONLY

40 track single sided	$165
80 track (dual sided 40 track)	$CALL
160 track (dual sided 80 track)	$CALL
8 inch Slimline sgl/dbl sided	$CALL
Winchester Hard Drives 5 Meg.	$399

CALL IF YOU FIND A LOWER PRICE ON DRIVES

CORVUS

HARD DISK DRIVES Complete from $1649

Single and multiuser HARD DRIVES for all brands of computers
One or several computers can share A HARD DISK.

Model 100 Portable Computer

8K / 24K / 32K Models

- Full-size Typewriter Keyboard
- 8 Line, 40 Character Display
- Weighs less than 4 pounds
- Battery operated with optional AC Power Adapter
- Built-in Word Processing
- Schedule and Address Book
- Built-in Modem
- Advanced BASIC Language
- RS232C Interface to printer/cassette recorder/phone
- Complete Documentation

ACCESSORIES

- 8k RAM Memory Expansion
- Parallel Printer Cable
- Direct-connect Modem Cable
- System Briefcase 20x14x4"
- AC Adapter

$CALL

59

TCS Heart of TEXAS COMPUTER SYSTEMS

P.O. Box 1327 Arlington, Texas 76004-1327

Toll Free 1-800-433-5184 Texas 1-817-274-5625

Payment by cashier's check, money order or certified check. Call for credit card $
No tax out of state. Texans add 5%. Prices subject to change at any time

Listing 3 continued

```
        LD      A,0C3H          ;PATCH SPRINKLER
        LD      (WATER),A       ;  SYSTEM TO
        LD      HL,AUTORT       ;    THE CLOCK
        LD      (WATER1),HL     ;      PROGRAM
        EI                      ;ENABLE INTERRUPTS
        RET
;*********************************************
;  MANUEL ROUTINE PATCHED TO CLOCK PROGRAM
;*********************************************
MANRT   LD      HL,SPRFLG       ;GET CONTROL WORD
        BIT     1,(HL)          ;MAKE SURE MANUEL MODE
        JR      Z.EXITMN        ;SKIP IF NOT SET
        BIT     2,(HL)          ;SET FOR ON-TIME COUNTDWN
        JR      NZ.SKIP         ;SKIP UNTIL TIME IS UP
        CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,(MNPORT)      ;LOAD ZONE ACTIVATE BIT
        OUT     (1H),A          ;TURN ON ZONE
        CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        LD      HL,SPRFLG       ;GET CONTROL WORD
        SET     2,(HL)          ;SET TO BYPASS ACT CODE
SKIP    LD      HL,(TENS)       ;LOAD SPRINKLER ZONE
        DEC     HL              ;  ON TIME & COUNT DWN
        LD      (TENS),HL       ;PUT TIME BACK
        LD      A,H             ;CHECK FOR
        OR      L               ;  TIME EQUAL
        CP      0               ;    TO ZERO
        JP      NZ.TOCLK        ;JUMP TO CLOCK
        CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,(MNPORT)      ;GET ZONE ACT BIT
        OUT     (1H),A          ;TURN OFF ZONE
        CALL    DLY             ;DELAY FOR RELAY TO LATCH
        JR      Z.EXITMN        ;SKIP IF ZONE TURNED OFF
        JP      ERROR           ;ERROR...ZONE STILL ON
EXITMN  CALL    AUPATC          ;PATCH AUTO & CLR FLAGS
        LD      HL,MESS11
        CALL    21BH            ;PRINT AUTO MODE MESS
        CALL    SOUND           ;ACKNOWLEGE MODE SET
        JP      TOCLK           ;JUMP TO CLK
;*********************************************
;  AUTO SPRINKLER MODE PATCHED TO CLK PROGRAM
;*********************************************
AUTORT  LD      HL,SPRFLG       ;GET CONTROL WORD
        BIT     7,(HL)          ;SET = SPRINKLER ACTIVE
        JR      NZ.AUT          ;BYPASS TIME CHECK
        LD      A,(AMPM)        ;READ A OR P FROM CLK
        CP      41H             ;IS IT AM?
        JP      NZ.TOCLK        ;LEAVE--NOT TIME TO START
        LD      A,(DIGIT)       ;GET DAY CODE FROM CLK
        CP      1               ;IS IT MONDAY?
        JR      Z.KK            ;SKIP AND START IF IT IS
        CP      3               ;IS IT WEDNESDAY?
        JR      Z.KK            ;SKIP AND START IF IT IS
        CP      5               ;IS IT FRIDAY?
        JR      Z.KK            ;SKIP AND START IF IT IS
        CP      8               ;THIS IS A SPARE DAY
        JP      NZ.TOCLK        ;LEAVE - NOT TIME TO STRT
KK      XOR     A               ;CLR CARRY FLG
        LD      HL,(HOURS)      ;GET ACTUAL TIME
        LD      BC,(HRS)        ;LOAD STRT TIME IN HRS
        SBC     HL,BC           ;ARE THEY EQUAL?
        JP      NZ.TOCLK        ;LEAVE AND KEEP TESTING
        XOR     A               ;CLR FLAGS
        LD      HL,(MINUTS)     ;GET ACTUAL MINUTES
        LD      BC,(MIN)        ;LOAD STRT TIME IN MIN
        SBC     HL,BC           ;ARE THEY EQUAL?
        JP      NZ.TOCLK        ;LEAVE AND KEEP TESTING
        LD      HL,SPRFLG       ;GET CONTROL WORD
        SET     7,(HL)          ;SET TO BYPASS TIME CHKS
AUT     LD      HL,SPRFLG       ;GET CONTROL WORD
        BIT     0,(HL)          ;TEST FOR AUTO ENABLED
        JP      Z.EXITMN        ;LEAVE IF AUTO NOT SET
        BIT     4,(HL)          ;SET FOR PAUSE DELAY
        JR      NZ.DELAY        ;SKIP FOR DELAY
        BIT     3,(HL)          ;ON TIME DELAY...ZONE ON
        JR      NZ.SKIP1        ;SKIP IF ZONE RUNNING
HH      CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,(CYCLE)       ;GET ZONE ACTIVATE BIT
        OUT     (1H),A          ;TURN ON ZONE
        CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        LD      HL,SPRFLG       ;GET CONTROL WORD
        SET     3,(HL)          ;SET TO SKIP HH SECTION
        LD      HL,ZHOLD        ;GET ZONE #
        INC     (HL)            ;INCREASE ZONE # BY 1
SKIP1   LD      HL,(TMEWRD)     ;GET ZONE ON TIME
        DEC     HL              ;DEC ON TIME ONCE/SECOND
        LD      (TMEWRD).HL     ;STORE TIME BACK
        LD      A,H             ;CHECK FOR
        OR      L               ;  TIME EQUAL
        CP      0               ;    TO ZERO
        JP      NZ.TOCLK        ;SKIP IF NOT
        LD      HL,RUNTME       ;GET ZONE ON TIME VALUE
        LD      (TMEWRD).HL     ;RESTORE ON TIME PERIOD
        CALL    BUSON           ;TURN ON INTERNAL BUS
        LD      A,(CYCLE)       ;GET ZONE ACTIVATE BIT
        OUT     (1H),A          ;TURN OFF ZONE
        CALL    DLY             ;TIME FOR RELAY TO LATCH
        JP      NZ.ERROR        ;ERROR...RELAY STILL ON
        CALL    BUSOFF          ;TURN OFF INTERNAL BUS
        LD      HL,SPRFLG       ;GET CONTROL WORD
        RES     3,(HL)          ;RESET TO ACT NEXT ZONE
        SET     4,(HL)          ;SET FOR PAUSE DELAY
DELAY   LD      HL,PAUSE        ;LOAD PAUSE TIME
        DEC     (HL)            ;DECREMENT PAUSE TIME
        XOR     A               ;CLR FOR COMPARE
        CP      (HL)            ;IS TIME = 0
        JP      NZ TOCLK        ;JUMP TO CLK IF NOT
        LD      HL,SPRFLG       ;GET CONTROL WORD
        RES     4,(HL)          ;RESET PAUSE DELAY
        LD      A,DLYTME        ;GET DELAY TIME VALUE
        LD      (PAUSE),A       ;RELOAD PAUSE TIME
        LD      A,(CYCLE)       ;GET ACTIVATE BIT
        SLA     A               ;SHFT LEFT = NEXT ACT BIT
        BIT     4,A             ;IF SET = AUTO MODE DONE
        JR      NZ.C1           ;CHECK FOR 2ND CYCLE
        LD      (CYCLE).A       ;PUT BACK SHIFTED BIT
        JP      TOCLK           ;GO FOR NEXT ZONE
;*********************************
;  ROUTINE TO CHECK FOR 2ND CYCLE
;*********************************
C1      LD      A,(CYK)         ;LOAD CYCLE COUNT
        DEC     A               ;DEC CYCLE
        CP      0               ;IS THIS 2ND PASS?
        JP      Z.EXITMN        ;SPRINKLER SYSTEM DONE
        LD      (CYK),A         ;LOAD DEC CYCLE # BACK
        LD      A,1             ;RESET SYSTEM TO ZONE #1
        LD      (CYCLE).A       ;  FOR SECOND PASS
        XOR     A
        LD      (ZHOLD),A       ;CLEAR ASCII ZONE #
        LD      A,'2'
        LD      (ZCYK),A        ;PUT IN MESS9 FOR PRINT
        JP      TOCLK           ;JUMP TO CLOCK
;*************************
;          VARIABLES
;*************************
HRS     DEFW    3530H           ;5 0   = TURN ON TIME
MIN     DEFW    3030H           ;0 0   = 05:00  AM
MNPORT  DEFB    0               ;Z CNTRL BIT FOR MAN
SPRFLG  DEFB    0               ;SPRINKLER CONTROL WORD
CYCLE   DEFB    1               ;SHIFTING ZONE ACT BIT
TMEWRD  DEFW    30              ;ZONE ON-TIME=15 MINUTES
PAUSE   DEFB    10              ;ZONE PAUSE TIME= 1.5 MIN
ZHOLD   DEFB    0               ;ZONE VALUE FOR ASCII PRNT
TENS    DEFB    0               ;USED FOR MAN TIME INPUT
UNITS   DEFB    0               ;" " " " "  TIME INPUT
CYK     DEFB    2               ;CYLE CONTROL FOR SYSTEM
;****************************************
;  THIS TABLE IS THE ACTIVATE BIT TABLE
;****************************************
PORT    DEFB    0H              ;DUMMY VALUE
        DEFB    1H              ;ZONE 1 ACTIVATE BIT
        DEFB    2H              ;ZONE 2
```

Listing 3 continued

Listing 4 continued

```
        CPL                     ;INVERSE VALUE FOR LENGTH
        DEC     A               ;DEC CMND LENGTH BY ONE
        LD      (CTR),A         ;SAVE LENGTH OF COMMAND
        LD      (DWNCTR),A      ;SAVE LENGTH OF COMMAND
        XOR     A               ;CLR A REGISTOR
        LD      B,A             ;CLR B REGISTOR
;****************************
;  COMMAND SEARCH SECTION
;****************************
        LD      HL,TABLE        ;GET STRT OF CUST CMD TBL
        LD      DE,42E6H        ;STRT OF BASIC INPUT BUFF
        LD      A,(ENDTBL)      ;GET LENTH OF COMMAND TBL
        LD      C,A             ;STORE LENGTH TO LOOP CTR
GO      LD      A,(DE)          ;FIRST BYTE OF BASIC CMD
        CP      0               ;CHECK FOR A CMD PRESENT
        JR      Z.RETURN        ;NO CMD...BACK TO BASIC
LOOP    CPI                     ;COMPARE BASIC AND TABLE
        JR      Z.MAYBE         ;SKIP TO MAYBE IF MATCH
        PUSH    AF              ;SAVE CONTENTS
        LD      A,(CTR)         ;LOAD LENGTH OF COMMAND
        LD      (DWNCTR),A      ;RESTORE DWN LOOP CTR
        LD      DE,42E6H        ;RET PTR TO BASIC CMD
        LD      A,(DE)          ;LOAD 1ST CHARCT & SEARCH
        POP     AF              ;RESTORE CONTENTS
INLOOP  JP      PE,LOOP         ;LOOP IF CTR NOT = 0
;*******************************************
;  RESTORE REGISTORS AND RETURN TO BASIC
;*******************************************
RETURN  POP     AF              ;RESTORE REGISTORS
        POP     BC
        POP     DE
        POP     HL
        RET                     ;RET TO CALLER IN BASIC
;*************************************************
;  MATCH WAS FOUND...CHECK NEXT CHARCT FOR MATCH
;*************************************************
MAYBE   PUSH    AF              ;SAVE CHARACTOR & FLAGS
        PUSH    HL              ;SAVE TABLE LOCATION
        LD      HL,DWNCTR       ;GET DOWN COUNTER
        DEC     (HL)            ;DEC DOWN COUNTER
        XOR     A               ;CLR A FOR COMPARE
        CP      (HL)            ;IS DOWN COUNTER = 0?
        JR      NZ.GOAGN        ;SKIP AND CHK NEXT CHARCT
        POP     HL              ;RESTORE TABLE LOCATION
        LD      A,(HL)          ;FOUND MATCH...GET NEXT
        CP      0C3H            ;  CHARCTOR AND COMPARE
        JR      Z.DONE          ;DONE = COMMAND MATCH
        POP     AF              ;RESTORE CHARCT & FLAGS
        JR      RETURN          ;SEARCH OVER... EXIT PROG
GOAGN   POP     HL              ;RESTORE TABLE LOCATION
        INC     DE              ;INC TO NEXT BASIC CHARCT
        POP     AF              ;RESTORE FLAGS
        LD      A,(DE)          ;GET NEW BASIC CMD CHARCT
        JR      INLOOP          ;GO TEST FOR COMPARE
;
DONE    JP      (HL)            ;JUMP TO COMMAND ADDRESS
;
;*************
;  VARIABLES
;*************
BEGIN   DEFW    START           ;START OF CUST CMD PROG
CTR     DEFB    0               ;LENGTH OF BASIC COMMAND
DWNCTR  DEFB    0               ;USED TO DEC CMD FOR TEST
;*************************
;  CUSTOM COMMAND TABLE
;*************************
TABLE   DEFM    'SPRINK'        ;SPRINKLER SYSTEM COMMAND
        DEFB    0C3H            ;JUMP ADDRESS TO
        DEFW    71E0H           ;  SPRINKLER SYSTEM
        DEFM    'ALARM'         ;ALARM SYSTEM COMMAND
        DEFB    0C3H            ;JUMP ADDRESS TO
        DEFW    7910H           ;  BURGLER ALARM PROGRAM
ENDTBL  DEFB    $-TABLE         ;CALC END OF CMD TABLE
;
        END     START
```

LOAD 80

Chameleon Code

by Bradley Murray

Tired of converting Assembly code into Basic data statements by hand? This speeds it up.

The mixture of Basic programs with Assembly-code subroutines is powerful, taking advantage of the simplicity of one and the speed of the other.

Sometimes the Assembly-code routines are loaded in Basic; other times it is better to POKE the decimal equivalents of the routine's object code into memory. It is a nuisance, though, to convert the object code to decimal, check the values and type the data lines.

I wrote a program that interprets an object file produced by an editor/assembler, converts the object code to decimal values, and writes these values to an ASCII file of data lines. These are later merged into the Basic program that POKEs the values into memory.

The program gives the starting and ending POKE locations, the memory size required to protect the POKEd program, and the hexadecimal (hex) address needed for a DEFUSRn = statement.

How It Works

Program Listing 1 requests the name of the file to be converted, the name of the ASCII data file (the output file), the starting line number of the data file, and the line number increment.

The program is listed one statement per line to increase its intelligibility, but when entering the program ignore all comment lines and compress statements into single lines. The following line numbers are branch points—you must preserve 1100, 1260, 1360, 1630, 1800, 1870, 1940, 2000, 2090, and 2240.

The program begins with an error trap: The object file first opens as an I-file, closes and then reopens as an R-file. If the file does not exist, a random file with no data is created and read, giving unpredictable, incorrect output. If the file is first opened as "I", the file-not-found error can be trapped.

Reading the file as an Input (I) file, with an INPUT #2, A$ would be simpler, but has one problem. A byte containing 00, valid in the object code, would be bypassed. The code D0A7 00FF would be incorrectly read into the string as D0A7FF. This does not happen when the File is read with a GET1,r (see line 1240).

Line 1090 sets HX$ to contain all the hex codes, from which values will be extracted in line 2070 to make a decimal-to-hex conversion.

Line 1180 fields the input file buffer as two variables of 128 bytes each. No single string variable can contain the full 256 bytes. If your older version of Basic has not been modified, the second half of the buffer field contains only 127 bytes. This leads to problems if your object program occupies a full disk sector or more (256 bytes).

The first byte of an editor/assembler object file is either 01 (if created by the EDTASM in the NEWDOS package), or 05 (if created by Radio Shack's Editor/Assembler). In either case the first byte of interest to this program contains 01—the command to load.

Line 1260 extracts a byte from the string A$(1). The first time through, it extracts the first byte and converts it to decimal. If it is not 01, the second byte will contain a certain number of bytes to be bypassed before loading program code. Line 1310 increases the pointer by this amount, to skip over the required bytes. If the following byte is 01, it's a loading instruction.

The byte following the loading instruction gives the number of bytes to be loaded, including 2 extra bytes containing the loading address. Lines 1380–1390 decode the number of program bytes NB to be extracted from the string. Zero is interpreted as 256, and 2 is subtracted from NB to take care of the loading-address bytes.

The loading address is used to calculate the first location into which Basic will POKE the program, as well as the memory size needed to protect the program.

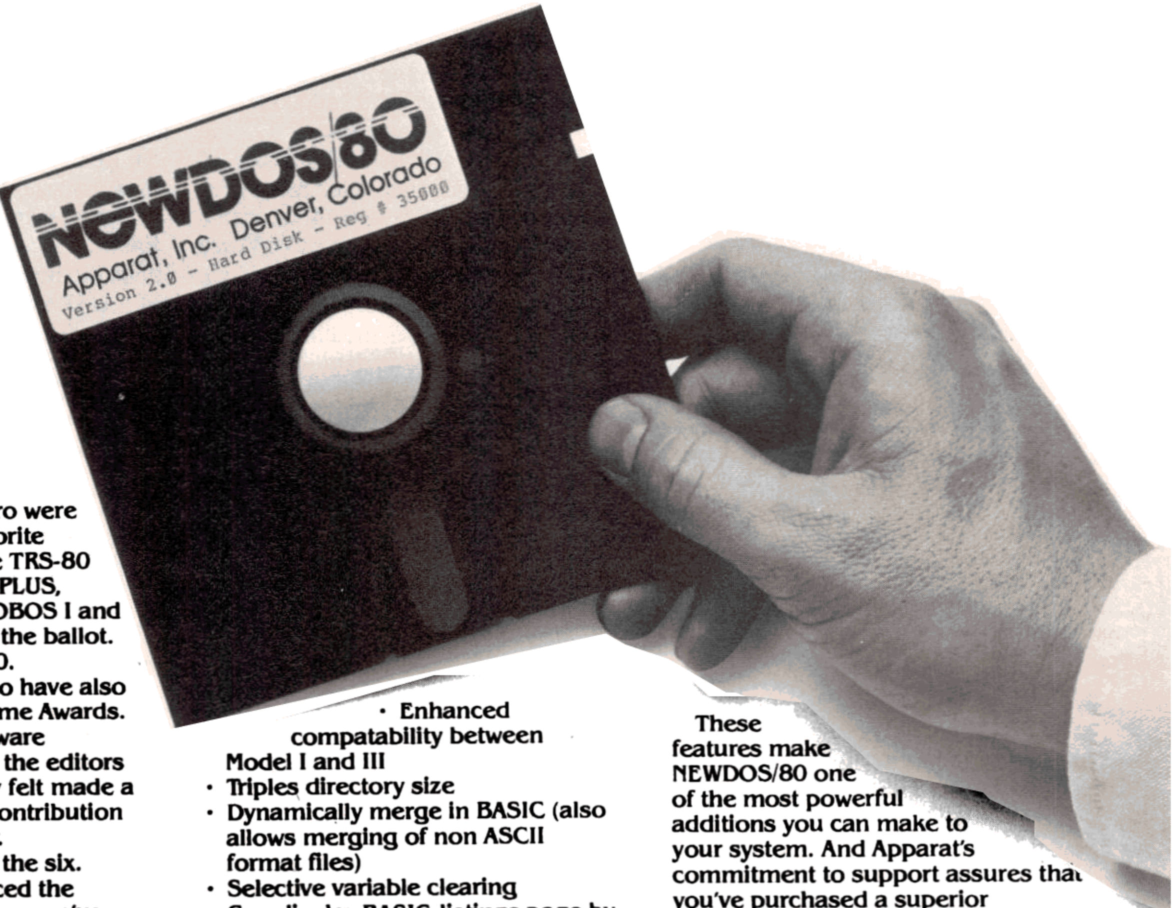
Line 1400 moves the pointer to the least significant bit (LSB) of the loading address and blanks the hex value string (ST$) for this address. The program calls subroutine 2000 to get the decimal value of the byte, find the quotient and remainder (modulo-16), and convert each to one of the hex characters in HX$. Upon return, it assigns ST the decimal value of LSB.

The pointer moves to the next byte, the most-significant bit (MSB) of the starting address. With a call to 2000, this is converted to decimal and to hex. The hex code is put before the LSB and linked with it to give the starting address in hex (ST$).

The Key Box

Model I
32K RAM
Disk Basic
Editor/Assembler

IF YOU'RE GOING TO BE PICKY ABOUT AN OPERATING SYSTEM SEE WHICH WAS PICKED BEST.



The readers of 80 Micro were asked to select their favorite operating system for the TRS-80 Model I&III. LDOS, DOSPLUS, TRSDOS, MULTIDOS, WOBOS I and NEWDOS/80 were all on the ballot. They picked NEWDOS/80.

The editors of 80 Micro have also awarded their Hall of Fame Awards. From among every software package on the market, the editors picked only six that they felt made a lasting and significant contribution to the TRS-80 computer. NEWDOS/80 was one of the six.

Since we first introduced the NEWDOS operating system we've been stating its features, capabilities and advantages. Thank you 80 Micro readers and NEWDOS/80 users for supporting us.

Version 2.0 . . .
High Performance DOS

NEWDOS/80 Version 2.0 is our highest performance system yet. The versatility and sophistication of Version 2.0 includes features like:

- Double density support on the Model I
- Enhanced compatability between Model I and III
- Triples directory size
- Dynamically merge in BASIC (also allows merging of non ASCII format files)
- Selective variable clearing
- Can display BASIC listings page by page
- Automatic repeat function key
- Routing for peripheral handling
- Enhanced disassembler
- Command chaining
- Superzap to scan files
- Fast sort function in BASIC

Hard Disk Support Now Available

- Support for Apparat's and Radio Shack's Model III hard disk (optional-available upon request for additional $60)

These features make NEWDOS/80 one of the most powerful additions you can make to your system. And Apparat's commitment to support assures that you've purchased a superior product, both today and tomorrow. At just $149.00 it could be the best investment you will make for your TRS-80.

For more information see your local computer store or contact Apparat, Inc., 4401 S. Tamarac Parkway, Denver, CO 80237, 303/741-1778.

TRS-80 and TRSDOS are registered trademarks of Tandy Corp., LDOS – Logical Systems, DOSPLUS – Micro Systems Software, MULTIDOS – Cosmopolitan Electronics, WOBOS I – Western Operations, NEWDOS/80 – Apparat.

Apparat, Inc.

✓40

Upon return from line 2000, the program multiples the decimal value of this byte by 256 and adds it to the LSB to give the starting address in decimal (ST). If the decimal value is greater than 32767, line 1520 sets the sign bit so the location to POKE will be negative. Line 1510 saves the positive value in MM as the memory size value. EN in line 1570 is the ending POKE address.

N points to the byte within the string A$(K). The subscript K is either 1 or 2; each record is split into two sub-records (see line 1180). K starts with a value of 1 in line 1620, increased when 128 bytes are decoded (lines 1730–1740). If K equals 3, 256 bytes have been decoded and the next record is read from the disk (1770–1780).

NB is the number of program bytes to be converted to decimal and put on the data line. KB is the number of bytes actually converted and added. Line 1800 compares NB and KB.

NN keeps track of the number of values on a single data line. As long as this value does not exceed 14, a comma is appended to the data line 1850. When there are 14 items, the routine writes a carriage return/line feed to the file (CHR$(13)), increases the line number, and generates a new data line (lines 1870–1890).

The decimal value of the code is not written directly to the data line, since spaces appear before and after the number. Lines 1630–1670 convert it to string format, trim it, and write it as a string.

When line 1800 determines that the number of bytes decoded and written (KB) equals the number of bytes required (NB), the program calls subroutine 2090, the ending routine. If the next byte is 01, more program bytes must be added to the data. NB is calculated as before, but the loading address is not needed.

If the next byte is not 01, the entire program has been loaded. Ordinarily this byte and the next would be 0202, followed by the program entry address (not always the loading address). In this conversion I assumed the starting address and the loading address were the same. If not, adjust them with a JP or JR to the starting address.

You can modify the program to handle a different starting address. If the next byte is not 01 (line 2180), check whether it and the next byte have a value of 02. If so, the 2 bytes following 0202 contain the entry address. Convert ST$ to this, using the technique above. ST (the start of loading) and MM (the memory size) remain unchanged.

If the byte is not 01 the loading is finished. The program branches to produce the hex value used for the address of a USRn call, the starting and ending loop index values for the POKE, and the memory size used in Basic when running the program.

Program Listing 2 POKEs the Assembly-code routine into memory and runs it. ■

Contact Bradley Murray, S.J., at Loyola College, Baltimore, MD 21210.

Program Listing 1. The Converter program.

```
1000 '            **  CONVERTER PROGRAM  **
1010 '
1020 '           CONVERTS AN OBJECT CODE FILE
1030 '           TO "DATA" LINES SUITABLE FOR
1040 '              BASIC "POKE" FUNCTION
1050 '
1060 '           TRAP FILE-NOT-FOUND ERROR
1070 ON ERROR GOTO 1940
1080 DIM A$(2)
1090 HX$="0123456789ABCDEF"
1100 LINEINPUT"OBJECT CODE FILESPEC: ";F$
1110 '          OPEN "I" TO SEE IF FILE EXISTS
1120 OPEN"I",1,F$
1130 CLOSE 1
1140 '          IF SO, OPEN AS A RANDOM FILE
1150 OPEN"R",1,F$
1160 '          BE SURE YOU HAVE UPDATED VERSION OF
1170 '           OF BASIC.  IF NOT: 127 AS A$(2)
1180 FIELD#1,128 AS A$(1),128 AS A$(2)
1190 LINEINPUT"ASCII DATA FILESPEC: ";F$
1200 OPEN"O",2,F$
1210 INPUT"ASCII DATA FILE STARTING LINE NUMBER ";LN
1220 INPUT"LINE NUMBER INCREMENT";IN
1230 PRINT #2,LN;"DATA ";
1240 GET 1,1
1250 N=1
1260 C=ASC(MID$(A$(1),N,1))
1270 '          C = LOAD COMMAND? (01H)
1280 '          IF NOT, NEXT BYTE IS NUMBER OF BYTES
1290 '          TO SKIP OVER.
1300 IF C=1 THEN 1360
         ELSE N=ASC(MID$(A$(1),N+1,1))
1310 N=N+3
1320 '          CHECK WHETHER THIS BYTE IS 01H
1330 GOTO1260
1340 '          YES: NEXT BYTE GIVES NUMBER OF BYTES TO
1350 '          LOAD, INCLUDING A TWO-BYTE LOADING ADDR
1360 N=N+1
1370 '           NB = NUMBER OF PROGRAM BYTES TO LOAD
1380 NB=ASC(MID$(A$(1),N,1))
```

Listing 1 continued

Listing 1 continued

```
1390 IF NB=0 THEN NB=254
     ELSE NB=NB-2
1400 N=N+1
1410 '          ST$ = STRING FOR HEX CODE OF LOADING ADDR
1420 ST$=""
1430 GOSUB 2000
1440 '          ST WILL BE LSB OF STARTING ADDRESS
1450 ST=K
1460 N=N+1
1470 GOSUB 2000
1480 '          ST WILL BE DECIMAL VA\LUE OF STARTING ADDR
1490 ST=ST+K*256
1500 '          SAVE ST IN MM (MEMORY SIZE)
1510 MM=ST
1520 IF ST>32767 THEN ST=ST-65536
1530 '          EN = ENDING ADDRESS
1540 '          KB=NUMBER OF PRGRAM BYTES READ FOR THIS
1550 '             SERIES OF LOADS (UP TO NEXT 01 CODE)
1560 '          N=BYTE BEING DECODED IN A$(K)
1570 EN=ST-1
1580 KB=0
1590 N=N+1
1600 '          NN = NUMBER OF VALUES ON A "DATA" LINE
1610 NN=0
1620 K=1
1630 C=ASC(MID$(A$(K),N,1))
1640 C$=STR$(C)
1650 '          REMOVE LEADING BLANK FROM NUMBER
1660 C$=RIGHT$(C$,LEN(C$)-1)
1670 PRINT #2,C$;
1680 N=N+1
1690 KB=KB+1
1700 EN=EN+1
1710 '          HAVE 128 BYTES BEEN READ?  IF SO
1720 '          START ON 2ND 128 BYTES OF RECORD
1730 IF N<129 THEN 1800
     ELSE N=1
1740 K=K+1
1750 '          IF 2ND 128 BYTES HAVE BEEN READ,
1760 '          GET NEXT RECORD
1770 IF K=2 THEN 1800
     ELSE GET 1
1780 K=1
1790 '          HAVE ALL BYTES BEEN READ?
1800 IF KB>=NB THEN GOSUB 2090
1810 NN=NN+1
1820 '          ARE THFRE 14 NUMBER ON "DATA" LINE?
1830 '          IF SO, START A NEW LINE; IF NOT,
1840 '          ADD ","
1850 IF NN=14 THEN 1870
     ELSE PRINT #2,",";
1860 GOTO 1630
1870 NN=0
1880 LN=LN+IN
1890 PRINT #2,CHR$(13);LN;"DATA ";
1900 GOTO 1630
1910 '---------------
1920 '          ERROR:  OBJECT FILE NOT FOUND
1930 '
1940 IF ERR<>106 THEN ON ERROR GOTO 0
1950 CLS : PRINT"FILE NOT FOUND"
1960 RESUME 1100
1970 CLOSE
1980 '---------------
1990 '           CONVERT 2-BYTE NUMBER TO HEX
2000 '
2010 '          GET DECIMAL VALUE FROM STRING
2020 K=ASC(MID$(A$(1),N,1))
2030 '          QUOTIENT & REMAINDER, MOD-16
2040 Q=INT(K/16)
2050 R=K-Q*16
2060 '          SELECT HEX VALUE FROM STRING HX$
2070 ST$=MID$(HX$,Q+1,1)+MID$(HX$,R+1,1)+ST$
2080 RETURN
2090 '---------------
2100 '          CHECK WHETHER THERE ARE MORE BYTES TO
2110 '          TO BE READ--I.E. NEXT BYTE IS 01.
2120 '          IF SO, GET NUMBER OF BYTES, AND SKIP
2130 '          LOADING ADDRESS--I.E. ASSUME IT FOLLOWS
2140 '          SEQUENTIALLY.  IF NO MORE BYTES, THEN
2150 '          ENDING ROUTINE: PRINT CRITICAL VALUES
2160 '
2170 C=ASC(MID$(A$(K),N,1))
2180 IF C<>1 THEN 2240
     ELSE N=N+1
2190 NB=ASC(MID$(A$(K),N,1))
2200 IF NB=0 THEN NB=254
     ELSE NB=NB-2
2210 KB=0
2220 N=N+3
2230 RETURN
2240 CLOSE
2250 CLS
2260 PRINT"FOR USR FUNCTION:  DEFUSRN = &H";ST$
2270 PRINT CHR$(13);"POKE VALUES FROM ";ST;"TO ";EN
2280 PRINT CHR$(13);"MEMORY SIZE:";MM
2290 PRINT CHR$(13);"MERGE";CHR$(34);F$;CHR$(34);CHR$(13)
2300 END
```

```
1000 '          ** THE POKE PROGRAM **
1010 '
1020 DEFINT A-Z
1030 INPUT"LINE 1000 OKAY?  <ENTER> OR BREAK";Q$
1040 INPUT"START AND END ADDRESSES";S,E
1050 FOR I=S TO E
1060 READ N : POKE I,N
1070 NEXT I
1080 DEFUSR1=
1090 X=USR1(0)
1100 END
```

Program Listing 2. The POKE program.

Basic, Faster and Readable—Part IV

by John Corbani

(The first three parts of this series appeared in June, p. 104, July, p. 200, and November, p. 228.—Eds.)

The intelligent use of variables in your programs is essential to both program speed and readability. The type and number of variables and their sequence affects execution time. Creating mnemonic variables that are easy to read, remember, and understand is essential to good programming.

In this article I look at the Model I/III version of Microsoft Basic and describe how Basic puts variables into memory and how to use variables so your computer reads them faster.

A variable name represents either a number (numeric variable) or a sequence of printable, or control, characters (string variables). Basic uses variables in program statements to represent different values. The concept is the same as that used in algebra.

The formula for adding three quantities illustrates the use of variables. A+B+C=D uses A, B, and C to represent values you want added together, and specifies D to represent the answer.

If you assign A the value 2, B the value 3, and C the value 5, then D equals 10 in the above formula. If A= "B" and B="O" and C= "Y" then D equals "BOY". You can change the number or characters represented by these variables, run the variables through a formula, and obtain general solutions to problems.

Using variables in a Basic program can affect execution speed. Make sure you use them efficiently.

Numeric Variables

Three types of numeric variables exist: integer, single-precision, and double-precision. Integer variables describe whole numbers between −32768 and +32767. Basic processes these variables very quickly.

Single-precision variables consist of any six digits and can contain a decimal point. They take longer to process than integer variables.

Double-precision variables use 16 digits with a floating decimal point and take the longest time to process, so use them only when necessary.

A numeric variable's name consists of one or two alphanumeric characters, with only two restrictions: the first character must be a letter, and you cannot use Basic reserved key words (If, To, Or, and so on).

Basic converts lowercase characters to uppercase. This allows for 26 single letter variables (A–Z), 260 alphanumeric (A0–Z9), and about 650 alpha/alpha combinations (AA–ZZ).

This total of 936 variables applies to each of the three types, bringing the total possible numeric variables to 2,808.

You have the choice of defining groups of variables as a specific numeric type, or adding an identifier (suffix) to the variable name every time you use it. If you do not specify a numeric type, Basic assumes a single-precision variable.

Use the DEFINT, DEFSNG, and DEFDBL commands to predefine groups of numeric variables as integer, single-, or double-precision.

The punctuation characters % (integer), ! (single-precision), and # (double-precision) are suffixes that you can add to variables to define their type. The listing below illustrates the use of these definitions and suffixes.

```
10 DEFINT A–C: DEFDBL X
20 A = −3: B! = 30: BA = 10: X = ⅓: X1 = 1:
   X2 = 3: X3 = X1/X2: PRINT A; B!; BA:
   PRINT X; X3
30 D = 1.666E3: X3 = 1.7563421D−2: PRINT
   D; X3
```

Line 10 defines all simple variables (those without a suffix) starting with the letters A, B, and C as integer variables. All simple variables starting with X are double-precision. All other simple variables are, by default, single-precision.

Line 20 assigns values to four variables. It sets A to −3, an integer, and single-precision variable B to 30.0000. It also sets BA to the integer 10, and X to the double-precision value of .3333333432674408.

The computer divides constants in single-precision and converts them to double-precision or integers. If you want ⅓ to equal .3333333333333333, you have to type it in yourself or make your computer do the division with double-precision numbers.

Line 30 illustrates a shorthand method to enter data that comes in scientific notation. When you assign values to single-precision variables, you can type the number, an E, and the exponent.

In double-precision you use a D before the exponent. Basic sets the variable D to 1666.00 and the variable X3 to .017563421.

String Variables

String variables represent characters or sequences of characters. They can represent from zero to 255 characters.

The rules for naming string variables are the same as those for numeric variables. Names can be either letters, an alphanumeric combination, or two letters that aren't reserved words.

Reserve groups of names for string variables by using the DEFSTR function (see the listing below). You can also identify a string variable by placing a $ (read "string," not "dollar") after a variable name. Basic provides 936 string variables, bringing the total number of variables to 3,744.

```
40 DEFSTR F-H: G = "Good bye."
   : H = "Hello"
50 Y$ = "you all.": F = CHR$(191) + "  "
60 PRINT F + H + Y$ + G
```

Line 40 defines all variables starting with the letters F–H as string variables, and assigns sequences of characters to G$ and H$.

Line 50 assigns a string to Y$ and assigns character number 191 (a full white block) and two spaces to F$. Line 60 uses the strings to print a block cursor and two sentences.

Arrays

Use numeric and string variables to identify arrays, or lists of values. The 3,744 possible arrays are in addition to the individual variables. Each array has as many as 255 dimensions, and 65,535 elements per dimension.

You don't need to define arrays with fewer than 11 elements per dimension. You must dimension arrays with any element number above 10 before referencing the high element.

If you define a variable as a certain type, you must dimension the variable array after that point as the same type unless otherwise specified by a suffix.

```
70 DIM A(20,2): DIM G(26)
80 FOR A = 1 TO 20: FOR B = 1 TO 2:
   A(A,B) = A B: NEXT B,A
90 FOR A = 1 TO 26: G(A) = CHR$(A + 64):
   NEXT
```

Line 70 dimensions two arrays. A(20,2) is a two-dimensional integer numeric array. G(26) is a single-dimensional string array. Remember, line 10 defines A as an integer and line 40 defines G as a string variable.

Line 80 fills the numeric array with the numbers from 1–20 in one dimension and the square of the numbers in the other dimension. Line 90 fills the string array with the code for the capital letters from A–Z

Basic stores in memory all the variables and arrays specified in a program, along with the information they represent. Basic stores numeric variables just after the program that defines them. The storage process occurs as the program uses each variable.

The program stores variables in memory in the order in which it encounters them. Each name takes 3 bytes: 1 for the variable type, and 2 for the name. The value assigned to numeric variables immediately follows.

Basic stores integer values in the next 2 bytes, single-precision in the next 4 bytes, and double-precision in the next 8 bytes. The length of the string in 1 byte and the address of the first character of the string in the 2 bytes follows the string variable's type and name. The program stores the data itself in high memory.

Basic stores arrays in a list following the simple variables. Three bytes identify the type of variable and the array name. Two bytes identify the total number of elements, and 1 byte identifies the number of dimensions. Basic also allocates 2 bytes for each dimension to give the number of elements allowed per dimension. The program stores the data, one element at a time, starting with the first dimension.

You can also use long names for the variables. Basic looks at the last character to determine type, and ignores all characters after the first two unless they are reserved (key) words. Microsoft gives more variables than most programs can use.

Why all the confusion about variables? Don't blame Basic, blame Fortran and the misinterpretation of books written by and for Fortran programmers. Fortran begat Basic, and contained only a few integer variables, starting at I.

Fortran had no strings, and virtually no organized structure. It was one of the first high-level languages, and is still the language of choice in most state-of-the-art scientific applications.

Basic is an interpreter that uses much of Fortran's source code. It has greatly expanded the role and power of variables. It also includes Basic-added numeric and string types, and many of the old restrictions are gone. The literature could not keep up. Many of the current problems are due to human factors that have seldom been addressed.

Be careful not to use letters in situations that could cause confusion. Electronic connector manufacturers have noticed that people misread the letters G, I, O, and Q. G, O, and Q turn up as zero. I is a 1, and vice versa. Connector

See List of Advertisers on Page 291

```
10  A% = 10: BC$ = "HELLO"
```

V = VARPTR(A%)

Memory Address	Value	Description
V − 3	3	Integer type code
V − 2	0	First letter of name (none)
V − 1	65	Second character of name (A)
V	10	Low byte of the value
V + 1	0	High byte of the value

V + VARPTR(BC$)

Memory Address	Value	Description
V − 3	3	String type
V − 2	66	First letter of name (B)
V − 1	67	Second character of name (C)
V	6	Number of characters in string
V + 1	254	Least significant byte of address of first letter of string (H)
V + 2	121	Most significant byte of first letter of string (H)
		(H) is found at 254 + (121*256), or 30,976

Table 1. An example of integer and string storage in Basic.

```
10 DEFINT A, B, V; B = 1•0: V = 1•0: DIM A(10,2)
20 FOR A = 0 TO 10: A(A,0) = A + 1: A(A,1) = A + 21:A(A,2) = A + 41: NEXT
30 V = VARPTR(A(0,0))
40 FOR B = − 10 TO 50: LPRINT PEEK(V + B):NEXT
```

Memory Address	Value	Description
V − 10	2	Integer variable
V − 9	0	Name
V − 8	65	Name
V − 7	71	Number of elements
V − 6	0	Number of elements
V − 5	2	Number of dimensions
V − 4	3	Depth of dimension 2
V − 3	0	Depth of dimension 2
V − 2	11	Depth of dimension 1
V − 1	0	Depth of dimension 1
V	1	First element A(0,0)
V + 1	0	First element A(0,0)...
V + 22	21	Element A(0,1)
V + 23	0	Element A(0,1)...
V + 44	41	Element A(0,2)
V + 45	0	Element A(0,2)

Table 2. A two-dimensional integer array and associated values.

manufacturers use a 22-letter alphabet. That's the reason for the standard 44-pin connector.

I was the first of the few integers in Fortran, and since integers ran fast, they were used in most programming books as preferred variables. Everyone used them, and everyone had problems.

G is not too bad on screen or on fully formed character printers, but if you have a dot matrix printer, watch out. O is a disaster when used alone, and it's even worse when used as the second letter of a variable name.

Don't even consider Q unless the program context makes it unmistakable. QS for queen of spades in a poker program makes good sense, for instance.

I is just as bad. It makes a bad second letter, and can cause confusion when used alone. IN is a handy name for input data, and if the context is clear, I is a reasonable first letter.

The logical selection of variable names depends on the purpose of the program. Pick letters and combinations that form mnemonics recognizable to anyone familiar with the general subject area. X and Y are good names for variables that describe the X and Y plotting positions in a graphics package.

D or T makes sense for loop variables used for time delays. NM$ is great for the name string in a mailing list program.

Whatever you do, keep a variables list in alphabetical order as you write a program, and put arrays in a separate list. You need to know what the variables are, what they represent, and how they're used. Never use the same name for two types of numerical variables.

Use the fewest number of variables possible to do the job. Nothing is wrong with specifying some variables as utility variables and using them repeatedly. Initialize them every time you use them and you'll never have a problem.

Variables can change function as a program executes. Just list all functions in the variables list. Variables and arrays take up valuable memory, whether used once or 1,000 times, so their efficiency is important. Remember, you can never recover memory during program execution.

Fewer variables mean higher speed. Basic interprets programs character by character. When it encounters a variable, the interpreter goes to the top of the variables list and starts searching for the name.

Variables used frequently or in time sensitive routines should be at the top of the list. If the variable is number 100 on the list, the program reads 99 variable names, compares them to the desired name, and discards them every time you reference that variable. Moving the variable up to 50 on the list doubles the speed of obtaining the quantity represented by that variable.

Assign critical values right at the start of the program. Remember to store string and numeric variables in different lists. Store string and numeric arrays in a separate list, and assign values accordingly.

Sometimes you'll want to know the location of a variable in memory. The POKE command can change the contents of strings, for example, or a machine-language routine might need access to some data in memory.

VARPTR(X) can get you to the right spot, but it's a tricky function to use. VARPTR(X) returns an error message if X is undefined in the program. If the program has not yet encountered X during the execution, VARPTR returns the memory location of the variable as part of the source code. This information is useless for determining type of value of the variable.

If all simple variables have values, and you use VARPTR to find an array, any answer you get is subject to change and may be wrong.

Assigning memory space to simple variables after arrays have been defined requires moving the array to make room for the simple variable. Table 1 shows how Basic stores an integer and a string. Table 2 shows a two-dimensional integer array.

Floating point numbers start with the least significant byte at location V. The number representation when playing with binary floating point numbers is too complex to go into here. Zilog and Intel have programming guides that can help, for those who are interested. Radio Shack information to date is superficial at best.

That's enough on variables. In my next installment, I will take a look at Basic commands that let you get right into the hardware of your computer. ■

John Corbani's hobbies include programming, radio controlled model aircraft, sailing, railroading, skiing, and windsurfing. You can reach him at 2455 Calle Linares, Santa Barbara, CA 93109.

Take It Off

by David Engelhardt

You can gain extra space on TRSDOS 6.0 disks by killing protected utility files. Here's how.

Model 4 owners may need or want more space on their TRSDOS 6.0 system disks. Killing infrequently used system utilities yields the extra space, but TRSDOS 6.0 protects these utilities with a password. Before you can rearrange your disks, you must circumvent this protection. I'll show you how.

Figure 1 shows the directory listing of an unaltered 6.0 disk. The letter P in the attribute (Attr) column identifies protected files.

The Concept

Debug on TRSDOS 6.0 is more powerful than the Model III version. It includes a new set of features, including a disk Read/Write utility. By taking advantage of this utility in 6.0's revised Debug, you bring the system directory into memory, remove the password, and write the modified directory back to the disk.

```
Drive :0  TRSDOS60   40 Cyl, DDEN, Free =   43.50K /   180.00K,  Date 10-Mar-83
Filespec     MOD   Attr   Prot    LRL  #Recs    EOF  File Size   Ext     Mod Date
---------------------------------------------------------------------------------
BAR/BAS                   FULL    256      0      0      0.00K     0
CLICK/FLT           P     EXEC    256      2    170      1.50K     1     10-Mar-83
COM/DVR             P     EXEC    256      4     45      1.50K     1     10-Mar-83
COMM/CMD            P     EXEC    256     12      2      3.00K     1     10-Mar-83
CONV/CMD            P     EXEC    256      7     38      3.00K     1     10-Mar-83
FLOPPY/DCT          P     READ    256      3     35      1.50K     1     10-Mar-83
FORMS/FLT           P     EXEC    256      4     35      1.50K     1     10-Mar-83
HERTZ50/JCL               FULL    256      2    174      1.50K     1     10-Mar-83
KSM/FLT             P     EXEC    256      4     44      1.50K     1     10-Mar-83
LOG/CMD             P     EXEC    256      2     94      1.50K     1     10-Mar-83
MAILLIST                  FULL    256     64     87     16.50K     1     10-Mar-83
MAILLIST/DAT        ?     FULL    256      1    255      1.50K     1     23-Apr-83
MEMDISK/DCT         P     READ    256     12    149      3.00K     1     10-Mar-83
REPAIR/CMD          P     EXEC    256      3     88      1.50K     1     10-Mar-83
TAPE100/CMD         P     EXEC    256      9     72      3.00K     1     10-Mar-83
=================================================================================
                15 files out of  37 selected, Space =    42.00K
```

Figure 1. TRSDOS 6.0 directory listing.

```
6200   5E 03 53 28   00 42 4F 4F   54 20 20 20   20 53 59 53   ^.S(.BOOT    SYS
6210   F6 37 F5 9C   0F 00 00 02   FF FF FF FF   FF FF FF FF   .7..............
6220   5F 03 53 89   00 53 59 53   36 20 20 20   20 53 59 53   _.S..SYS6    SYS
6230   F6 37 F5 9C   30 00 16 07   FF FF FF FF   FF FF FF FF   .7..0...........
6240   15 03 53 96   00 4D 45 4D   44 49 53 4B   20 44 43 54   ..S..MEMDISK DCT
6250   BF AE 96 42   0C 00 0E 01   FF FF FF FF   FF FF FF FF   ...B............
6260   16 03 53 59   00 52 45 50   41 49 52 20   20 43 4D 44   ..SY.REPAIR  CMD
6270   BF AE 96 42   03 00 09 40   FF FF FF FF   FF FF FF FF   ...B...@........
6280   16 03 53 2E   00 43 4F 4D   20 20 20 20   20 44 56 52   ..S..COM     DVR
6290   51 1F 96 42   04 00 0C 20   FF FF FF FF   FF FF FF FF   Q..B... ........
62A0   16 03 53 49   00 54 41 50   45 31 30 30   20 43 4D 44   ..SI.TAPE100 CMD
62B0   BF AE 96 42   09 00 0A 01   FF FF FF FF   FF FF FF FF   ...B............
62C0   00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00   ................
62D0   00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00   ................
62E0   16 03 53 5F   00 4C 4F 47   20 20 20 20   20 43 4D 44   ..S_.LOG     CMD
62F0   BF AE 96 42   02 00 09 00   FF FF FF FF   FF FF FF FF   ...B............
```

Figure 2. Debug display.

Your TRSDOS 6.0 manual discusses this utility on pages 1-75. When you invoke a disk utility command, you must specify six parameters relevant to the operation that tell 6.0 how to proceed. You need to indicate the number of the disk drive, the cylinder number, the starting sector, the type of operation you want to perform, the memory address, and the number of sectors.

The Read/Write utility allows a possibility of four disk drives, 0–3.

TRSDOS 6.0's name for tracks is cylinders, and you must cite cylinder values in hexadecimal (hex) units. Therefore, the range of possible cylinders on a 40-track disk is 0–27 hex.

The starting sector indicates where you want to begin reading from the disk.

The Read/Write utility performs three operations: it writes to a disk (the W command), reads from a disk (R), and writes to the directory (*).

The address value tells the system where in memory to load the information read from the disk. Again, this value must be hexadecimal.

You control how much of a disk the system reads by specifying a number of sectors. If you don't specify, the program reads a whole cylinder (18 sectors). Since sectors are numbered 0–17, selecting a sector number greater than 17 tells the software to continue reading onto the next cylinder.

Removing the Protection

Begin the unprotecting process with a little insurance. Instead of altering your original disk, make a back-up copy and use that. If anything goes wrong while you're tampering with the disk directory, you won't be able to recover the working disk.

Place the back-up in drive zero. Type the command FREE :0 to determine which cylinder holds the directory. The directory location is labelled DDD, and is usually at cylinder 20.

At the TRSDOS Ready prompt, type DEBUG (ON,EXT) and press the enter key. The Debug monitor loads, then control passes back to the Ready prompt. Press the break key to enter the Debug monitor and a three-quarters screen display of the Z80 register values appears.

Type F6000,7800,00 and press the enter key. This clears the memory you want to use.

Load the disk's directory into memory using the Disk Read/Write utility command: 0,14,0,R,6000,17 and press the enter key. The software reads disk cylinder 14 hex and places the information in memory beginning at 6000 hex. Then the utility displays the contents of memory, starting at 6000 hex—this is the disk's directory.

Pressing the semicolon key increments the display 256 bytes of memory at a time; the hyphen key decrements the display by 256 bytes. Step through memory until the starting address of 6200 hex appears on the screen.

In the far right column, the file names appear in ASCII format. As you step through memory, the names of all the disk's files show up. Notice that some of these names don't appear when you take a directory of the disk. See your manual's explanation of the DIR command for more about these "invisible" files.

Figure 2 shows the Debug display of memory beginning at 6200 hex. This includes the directory information for the utilities Memdisk, Repair, Com, Tape-100, and Log. The line on which the ASCII file name appears represents the beginning of each directory entry.

Looking at the first byte in the first line of each utility, notice Memdisk begins with a 15; Repair, Com, Tape-100, and Log begin with 16. This byte tells the system the program's protection level. Changing this changes the program's protection. If the value is 10 hex, the program is completely accessible.

Try unprotecting Memdisk for practice. Type E6240 and press the space bar to set Debug's pointer to Memdisk's first line. Notice that the current byte value appears at the lower left of the screen.

Type 10 and press the enter key. The display reflects the change you made. The first byte of Memdisk is 10, meaning that Memdisk is now unprotected.

Type E6260 and press the space bar. The pointer moves to the beginning of the Repair utility. Change its initial byte to 10. Continue making these changes until you alter all the utility programs. Remember: You've made these changes in memory only, not on the disk.

To replace the disk's directory with your modified directory, use another disk Read/Write utility command. It's almost the same as the one that puts the directory in memory. Just change the operator from R to an asterisk. Type 0,14,0,*,6000,17 and press the enter key. Type the letter O and press the enter key to return to TRSDOS Ready. Turn off Debug using the command DEBUG (OFF).

Call the directory using the DIR command and you see that the programs you changed no longer have the P attribute. You are free to copy or kill them at your discretion.

Use this technique to customize your 6.0 disks to suit your needs.

(For related information, see the sidebar on the next page.—Eds.) ■

Contact David Engelhardt at 10221 W. 101st Place, Broomfield, CO 80020.

The Password Is . . .

by David Lantis

At one time or another, you've probably run across a Model 4 disk file you wanted to list but couldn't because it was password-protected. One way to gain access is to remove the password protection, but that leaves the files open to anyone.

A better solution is to use this Basic program and the Memdisk utility to uncover the password. This leaves your files protected against accidents, while still giving you complete freedom.

Program Operation

PASSCRCK/BAS (see the Program Listing) searches for the password of a protected file on the disk in drive zero. The program systematically tries every possible combination of letters and numbers until it hits upon the successful one.

The On Error and Resume statements in lines 10 and 5000, respectively, are PASSCRCK/BAS's key. They keep the program running as long as it makes errors, i.e., as long as it generates an incorrect password.

Lines 30–160 loop around, testing every possible alphanumeric value for each of seven possible password characters (stored in variables C2–C8). Line 170, the loop that tests the value of the first character (C1), is different from those in the previous lines because TRSDOS 6.0 requires that a letter begin a password.

Lines 180–200 tally the values of the individual characters (C1–C8) into P$, the variable that holds the password possibility currently being tested. Line 210 is optional; it prints P$ on the screen. While it's helpful to know how far along the program is, this line does slow execution.

Line 220 sets the variable FILE$ equal to the command that lists your selected program, LIST FILE NAME/EXTENSION PASSWORD, where password is the possibility the program tests.

Line 230 goes out to the system to test the password. If the program is successful, then the program listing appears on the screen. Finally, the software prints the line LIST FILE NAME/EXTENSION PASSWORD at the top of the screen. Get a pencil and write down the password.

Unfortunately, this program is slow. It tests tens of thousands of possible combinations of letters and numbers. In addition to the time it takes to test the password, the disk drives wait one second between the time they start spinning and the time they begin reading the disk. Multiply this second across the number of password possibilities tested and we're talking about hours, maybe days.

Speeding Things Up

Memdisk, a TRSDOS 6.0 utility, helps speed things by creating a pseudo-disk in memory. This circumvents the physical disk I/O that takes so much time. Your TRSDOS 6.0 manual documents the Memdisk utility on page A-116.

To run PASSCRCK/BAS with Memdisk, move the system files, Basic's error reporting overlay BASIC/OV1, and the program you want to crack into memory on Memdisk. (If BASIC/OV1 is protected on your disk, remove the protection using the method described above.)

Type SYSTEM (DRIVE = 2, DRIVER = "MEMDISK") and press the enter key. The Memdisk utility menu appears. Choose option D, specify a double-density Memdisk (type D again), and respond Y to the format request. If you ask for a directory (DIR), you see that drive 2 now contains Memdisk.

Type in BACKUP $/SYS:0 :2 (SYS) to tell the computer to copy all system files onto disk 2, the disk in memory.

Typing in BACKUP BASIC/OV1:0 :2 moves the Basic overlay program onto Memdisk. This circumvents the physical disk I/O involved when PASSCRCK/BAS fails and BASIC/OV1 reports, "Access denied due to password protection."

Type in BACKUP file/ext:0 :2 (where file is the name of the program you're cracking) to move the program onto Memdisk.

Typing in SYSTEM (SYSTEM = 2) installs Memdisk as drive zero. Requesting a directory now shows physical drive zero as drive 2 and Memdisk as drive zero.

Type in BASIC/CMD:2 to run Basic. Then, from Basic's READY prompt, load and run PASSCRCK/BAS. When the program asks "File you wish to list?" enter the name of the program you want to crack.

With the Memdisk method implemented, PASSCRCK/BAS tests about 100 passwords per minute. Carried to its ultimate end, the program reaches its final possibility (password ZZZZZZZZ) in approximately 1,041 years. But the chances are that the software will encounter the correct password well before then. ■

```
10 ON ERROR GOTO 5000
20 INPUT"What file to list -- must be on disk 0";F$
30 FOR C8=32 TO 90
40 IF (C8>32 AND C8<48) OR (C8>57 AND C8<65) THEN 320
50 FOR C7=32 TO 90
60 IF (C7>32 AND C7<48) OR (C7>57 AND C7<65) THEN 310
70 FOR C6=32 TO 90
80 IF (C6>32 AND C6<48) OR (C6>57 AND C6>65) THEN 300
90 FOR C5=32 TO 90
100 IF (C5>32 AND C5<48) OR (C5>57 AND C5<68) THEN 290
110 FOR C4=32 TO 90
120 IF (C4>32 AND C4<48) OR (C4>57 AND C4<68) THEN 280
130 FOR C3=32 TO 90
140 IF (C3>32 AND C3<48) OR (C3>57 AND C3<68)THEN 270
150 FOR C2=32 TO 90
160 IF (C2>32 AND C2<48) OR (C2>57 AND C2<68) THEN 260
170 FOR C1=65 TO 90
180 P1$=CHR$(C1)+CHR$(C2)+CHR$(C3)+CHR$(C4)
190 P2$=CHR$(C5)+CHR$(C6)+CHR$(C7)+CHR$(C8)
200 P$=P1$+P2$
210 PRINT P$
220 FILE$="list "+F$+"."+P$
230 SYSTEM FILE$
240 PRINT FILE$: END
250 NEXT C1
260 NEXT C2
270 NEXT C3
280 NEXT C4
290 NEXT C5
300 NEXT C6
310 NEXT C7
320 NEXT C8
330 PRINT "Couldn't find the password, I guess you'll have to fi
re me":END
5000 RESUME 250
```

Program Listing. Password Sleuth.

80 micro BOOK SHELF

PROGRAMMING•6809

SPECIAL INTERESTS

MICROSOFT BASIC DECODED AND OTHER MYSTERIES—by James Earvour. From the company that brought you *TRS-80 DISK AND OTHER MYSTERIES.* Contains more than 6500 lines of comments for the disassembled Level II ROMs, six additional chapters describing every BASIC subroutine, with assembly language routines showing how to use them. Flow charts for all major routines showing how to use them. Flow charts for all major routines give the reader a real insight into how the interpreter works. BK1186 $29.50.*

PROGRAMMING THE Z-80—by Rodnay Zaks. Here is assembly language programming for the Z-80 presented as a progressive, step-by-step course. This book is both an educational text *and* a self-contained reference book, useful to both the beginning and the experienced programmer who wish to learn about the Z-80. Exercises to test the reader are included. **BK1122 $16.95** *

ASSEMBLY LANGUAGE GRAPHICS FOR THE TRS-80 COLOR COMPUTER—by Don Inman and Kurt Inman with DYMAX. This dynamic new book uses sound and graphics to show you how 6809 assembly language can be used to perform tasks that would be difficult or impossible with BASIC. All of the techniques are explained in a hands-on approach. Learn how to tailor you own programming style, from editing, assembling, executing and even debugging, to making your own programs run quickly and efficiently. It is also packed with video screen diagrams which explain each step of the process of creating your own graphics. BK1277 $??.??

TRS-80 COLOR BASIC—Bob Albrecht. Learn how to use the unique color, sound and graphics of the TRS-80 Color Computer. This self-teaching guide uses a learn-as-you-play format to teach Color BASIC. Packed with games, experiments, programming problems, and solutions, it is an ideal introduction for children, teachers, and adults. It starts with simple concepts and takes you on to more complicated games, graphics, and activities, including many chances for you to try out your newly learned programming skills. Included is an entire chapter on programming problems, which offers tip on adapting to Microsoft BASIC on other personal computers. BK1280 $10.95

SO YOU ARE THINKING ABOUT A SMALL BUSINESS COMPUTER—By Richard G. Canning and Nancy C. Leeper. For a well-organized manual on the process of selecting the right computer system for your small business, this text can't be excelled. Designed to introduce the novice in data and word processing to the real benefits of computerization, the book is filled with money- and time-saving tips, photos of equipment, lists of suppliers, prices, explanations of computer terminology, and helpful references to additional sources of information. Everyone contemplating a first computer installation should have this book. BK1222 $14.00*

THE CUSTOM TRS-80 AND OTHER MYSTERIES—by Dennis Kitsz. More than 300 pages of TRS-80 customizing information. With this book you'll be able to explore your computer like never before. Want to turn an 8 track into a mass storage unit? Individual reverse characters? Replace the BASIC ROMs? Make Music? High speed, reverse video, Level I and Level II? Fix it if it breaks down? All this and much, much more. Even if you have never used a soldering iron or read a circuit diagram, this book will teach you how! This is the definitive guide to customizing your 80! BK1218 $29.95.*

6809 MICROCOMPUTER PROGRAMMING AND INTERFACING—by Andrew C. Staugaard, Jr. Getting involved with Tandy's new Color Computer? If so, this new book from the Blacksburg Group will allow you to exploit the awesome power of the machine's 6809 microprocessor. Detailed information on processor architecture, addressing modes, register operation, data movement, arithmetic logic operations, I/O and interfacing is provided, as well as a review section at the end of each chapter. Four appendices are included covering the 6809 instruction set, specification sheets of the 6809 family of processors, other 6800 series equipment and the 6809/6821 Peripheral Interface Adapter. This book is a must for the serious Color Computer owner. BK1215 $13.95.*

TRS-80 DISK AND OTHER MYSTERIES—by Harvard C. Pennington. This is the definitive work on the TRS-80 Model I disk system. It is full of detailed "How to" information with examples, samples and in-depth explanations suitable for beginners and professionals alike. The recovery of one lost file is worth the price alone. BK1181 $22.50.*

INSIDE LEVEL II—For machine language programmers. This is a comprehensive reference guide to the Level II ROMs, allowing easy utilization of the sophisticated routines they contain. It concisely explains set-ups, calling sequences, variable passage and I/O routines. Part II presents an entirely new composite program structure which unloads under the SYSTEM command and executes in both BASIC and machine code with the speed and efficiency of a compiler. Special consideration is given to disk systems. BK1183 $15.95.*

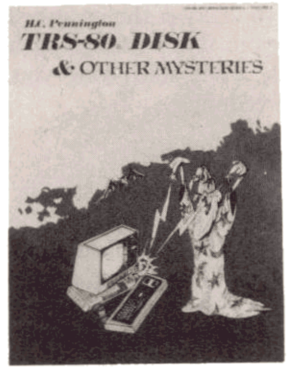
Nanos System REFERENCE CARDS

FOR MODELS I, II, III, COLOR AND POCKET COMPUTERS!

At last! No more flipping through the pages of the BASIC manual! No more working through the maze of machine language instructions! These cards completely summarize the BASIC and Assembler manuals! FEATURES INCLUDE:
memory map, eyeball graphics, math instructions, BASIC commands, store instructions, BASIC functions, load instructions, BASIC statements, move instructions, special keys, exchange instructions, PRINT USING examples, shift instructions, BASIC special characters, compare instructions, BASIC and assembler messages and codes, branch instructions, BASIC facts, data alteration instructions, reserved words, I/O instructions, ROM routines, complete character chart with graphics and space-compression codes, hex-dec chart, control code cross-reference, assembler instructions, commands and operators, screen line layout, editor commands and subcommands, condition code easy access.
Plus—"magic graphics number—a mystery until you learn how to use it!
Designed as a fold-up, accordion-style card, fits in your pocket. Panels organized for optimum speed for reference.

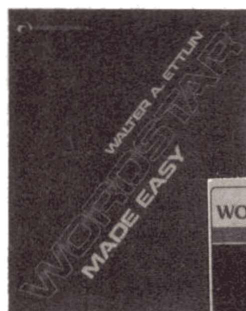
Model I: BASIC and Assembler FC1002	$4.95
Model II: BASIC and Assembler FC1005	$5.95
Model II: Commands and Utilities FC1010	$3.95
Model III: BASIC and Assembler FC1005	$5.95
Color: BASIC and Extended FC1006	$4.95
Z-80: Microprocessor FC1011	$4.95
ZX80, ZX81 and Timex Sinclair 1000: FC1012	$5.95
Pocket Computer: BASIC FC1009	$2.95
Apple II and Apple II Plus: BASIC and 6502 FC1008	$4.95
Apple II and Apple II Plus: BASIC only FC1007	$3.95

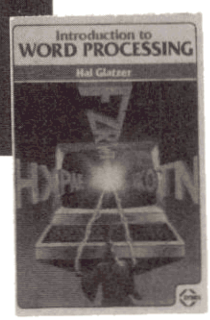
MASTERING CP/M—by Alan R. Miller For advanced CP/M users or systems programmers who want maximum use of the CP/M operating system, this book takes up where the *CP/M Handbook* leaves off. It will give you an in-depth understanding of the CP/M modules such as CCP (Console Command Processor), BIOS (Basic Input/Output System), and BDOS (Basic Disk Operating System). It explains the incorporation of additional peripherals to the system, console I/O, the use of the file control block and much more. It also includes a library of useful macros and a comprehensive set of appendices. BK1263 **$16.95**



A USER GUIDE TO THE UNIX SYSTEM by Jean Yates and Rebecca Thomas. Here at last is a clearly written book that allows you to use the Unix operating system easily, and at a fraction of the time it previously took. If you're using, evaluating or simply curious about this system, this is your book. BK1242 $15.99

WORDSTAR MADE EASY by Walter A. Ettlin. Now WordStar is as simple to learn as it is easy to use. This book teaches WordStar in 14 easy lessons, saving hours of hard work. It comes with a convenient pull-out Command Card. BK1239 $7.95



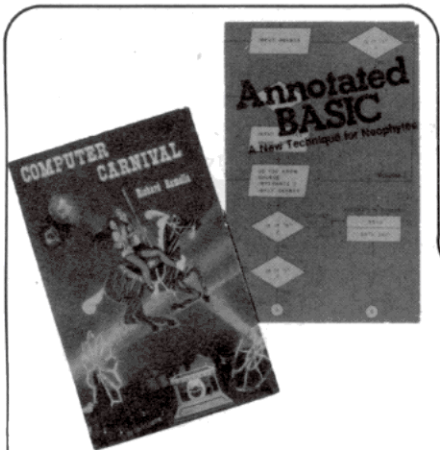


INTRODUCTION TO WORD PROCESSING by Hal Glatzer. This book explains in plain language what a word processor can do, how to use one, how it improves productivity—especially in businesses that handle lots of words—and how to buy one wisely. No technical knowledge required, for all first-time users and those considering purchasing a word processor. BK1238 $12.95

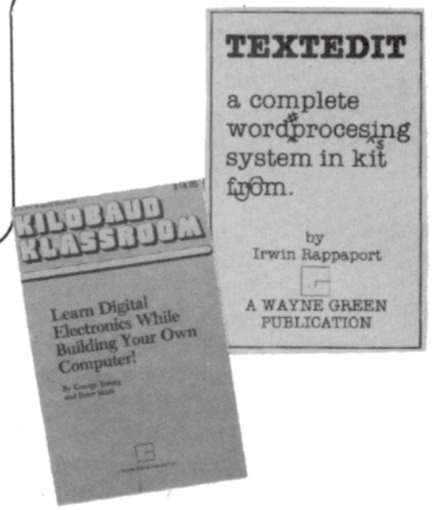
*Use the order card in this magazine or itemize your order on a separate piece of paper and mail to: 80 Micro Book Department • Peterborough NH 03458. Be sure to include check or detailed credit card information. No C.O.D. orders accepted. $1.50 for the first book, $1.00 each additional book for U.S. delivery and foreign surface. For foreign airmail $10.00 per book. Please allow 4-6 weeks for delivery. Questions regarding your order? Please write to Customer Service at the above address.

For Toll Free Ordering Call 1-800-258-5473

PRICES SUBJECT TO CHANGE WITHOUT NOTICE



80 micro BOOK SHELF



ANNOTATED BASIC—A NEW TECHNIQUE FOR NEOPHYTES—BASIC programming was supposed to be simple—a beginner's programming language which was so near to English that is could be easily understood. But, in recent years, BASIC has become much more powerful and therefore much more difficult to read and understand. BASIC simply isn't basic anymore.

Annotated BASIC explains the complexities of modern BASIC. It includes complete TRS-80* Level II BASIC programs that you can use. Each program is annotated to explain in step-by-step fashion the workings of the program. Programs are flowcharted to assist you in following the operational sequence. And—each chapter includes a description of the new concepts which have been introduced.

Annotated BASIC deals with the hows and whys of TRS-80 BASIC programming. *How* is a program put together? *Why* is it written that way? By observing the programs and following the annotation, you can develop new techniques to use in your own programs—or modify commercial programs for your specific use.
Annotated BASIC Volume 1 BK7384 $10.95
Annotated BASIC Volume 2 BK7385 $10.95
Order Both Volumes and Save! BK738402 $18.95

COMPUTER CARNIVAL—by Richard Ramella. Your child can become a crackerjack computerist with the sixty TRS-80 Level II programs in COMPUTER CARNIVAL. This large-type, spiral bound book for beginners is a veritable funhouse of games, graphics, quizzes and puzzles. Written by *80 Micro* columnist Richard Ramella, the programs are challenging enough to ensure continued learning, yet short enough to provide your child with the immediate delight and reward of mastering basic computing skills. And for even greater enjoyment, get the CARNIVAL COMPANION, a 30-minute cassette containing all the programs in the book. Eliminates tiresome typing and lets your child spend more time enjoying the programs. BK7389 $16.97
CC7389 Book and Cassette $24.97

THE NEW WEATHER SATELLITE HANDBOOK—by Dr. Ralph E. Taggart WB8DQT. Here is the completely updated and revised edition of the best-selling *Weather Satellite Handbook*—containing all the information on the most sophisticated and effective spacecraft now in orbit. Dr. Taggart has written this book to serve both the experienced amateur satellite enthusiast and the newcomer. This book is an introduction to satellite watching, that tells you how to construct a complete and highly effective ground station. Not just ideas, but solid hardware designs and all the instructions necessary to operate the equipment are included. An entire chapter is devoted to microcomputers and the Weather Satellite Station, and for the thousands of experimenters who are operating stations. *The New Weather Satellite Handbook* details all the procedures necessary to follow the current spacecraft. Weather Satellite contains Operation Satellite Systems, Antenna Systems, Weather Satellite Receivers, A Cathode Ray Tube (CRT) Monitor for Satellite Picture Display, A Direct-Printing Fascimile System for Weather Satellite Display, How to Find the Satellite, Test Equipment, Microcomputers and the Weather Satellite Station, Station Operations. BK7383 $8.95.*

THE SELECTRIC INTERFACE—by George Young. You need the quality print that a daisy wheel printer provides but the thought of buying one makes your wallet wilt. SELECTRIC™ INTERFACE, a step-by-step guide to interfacing an IBM Selectric I/O Writer to your microcomputer, will give you that quality at a fraction of the price. George Young, co-author of *Kilobaud Microcomputing* magazine's popular "Kilobaud Klassroom" series, offers a low-cost alternative to buying a daisy wheel printer. SELECTRIC INTERFACE includes: step-by-step instructions, tips on purchasing a used Selectric, information on various Selectric models, including the 2740, 2980, and Dura 1041, driver software for Z80, 8080, and 6502 chips, tips on interfacing techniques. With SELECTRIC INTERFACE and some background in electronics, you can have a high-quality, low-cost, letter-quality printer. Petals not included. BK7388 (125 pages) $12.97

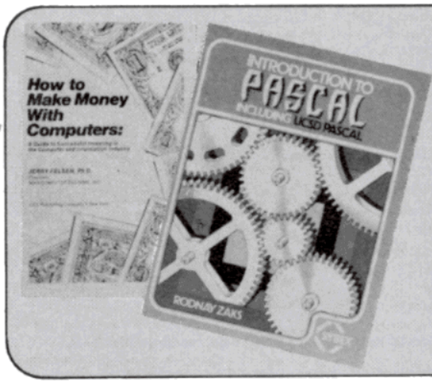
KILOBAUD KLASSROOM—by George Young and Peter Stark. Learning electronics theory without practice isn't easy. And it's no fun to build an electronics project that you can't use. *Kilobaud Klassroom*, the popular series first published in *Kilobaud Microcomputing*, combines theory with practice. This is a *practical* course in digital electronics. It starts out with very simple electronics projects, and by the end of the course you'll construct your own working microcomputer!

Authors Young and Stark are experienced teachers, and their approach is simple and direct. Whether you're learning at home or in the classroom, this book provides you with a solid background in electronics—and you'll own a computer that you built yourself! BK7386 $14.95

TEXTEDIT—A Complete Word Processing System in Kit Form—by Irwin Rappaport. TEXTEDIT is an inexpensive word processor that you can adapt to suit your differing needs—from form letters to lengthy texts. Written in TRS-80 Disk BASIC, the system consists of several modules, permitting the loading and use of only those portions needed. A disk is also available which provides the direct loading of the modules, however, the book is required for documentation. For Model I and III with TRSDOS CONVERT., one disk drive (2 disk drives or copy utility needed to transfer to system disk). Runs under TRSDOS 2.2/2.3. May not function under other systems. BK7387 $9.97 Disk DS7387 $19.97

40 COMPUTER GAMES FROM *KILOBAUD MICROCOMPUTING*—Forty games in all in nine different categories. Games for large and small systems, and even a section on calculator games. Many versions of BASIC used and a wide variety of systems represented. A must for the serious computer gamesman. BK7381 $7.95.*

WAYNE GREEN BOOKS



BASIC & PASCAL

HOW TO MAKE MONEY WITH COMPUTERS—In 10 information-packed chapters, Jerry Felsen describes more than 30 computer-related, money-making, high profit, low capital investment opportunities. BK1003 $15.00.*

INTRODUCTION TO PASCAL—By Rodnay Zaks. A step-by-step introduction for anyone wanting to learn the language quickly and completely. Each concept is explained simply and in a logical order. All features of the language are presented in a clear, easy-to-understand format with exercises to test the reader at the end of each chapter. It describes both standard PASCAL and UCSD PASCAL, the most widely used dialect for small computers. No computer or programming experience is necessary. **BK1189 $16.95**

PROGRAMMING IN PASCAL—by Peter Grogono. The computer programming language PASCAL was the first language to embody in a coherent way the concepts of structured programming, which has been defined by Edsger Dijkstra and C.A.R. Hoare. As such, it is a landmark in the development of programming languages. PASCAL was developed by Niklaus Wirth in Zurich; it is derived from the language ALGOL 60 but is more powerful and easier to use. PASCAL is now widely accepted as a useful language that can be efficiently implemented, and as an excellent teaching tool. It does not assume knowledge of any other programming language and therefore suitable for an introductory course. BK1140 $12.95.*

*Use the order card in this magazine or itemize your order on a separate piece of paper and mail to: 80 Micro Book Department • Peterborough NH 03458. Be sure to include check or detailed credit card information. No C.O.D. orders accepted. $1.50 for the first book, $1.00 each additional book for U.S. delivery and foreign surface. For foreign airmail $10.00 per book. Please allow 4–6 weeks for delivery. Questions regarding your order? Please write to Customer Service at the above address.

For Toll Free Ordering Call 1-800-258-5473

PRICES SUBJECT TO CHANGE WITHOUT NOTICE

Finding the Search Solution

Continued from p. 115

of the two keys and uses the composite key signature for the search. For example, to search for the record containing the keys "lane" and "walk", the program logically ORs the key signatures of "lane" and "walk" to produce a composite key signature (Fig. 4).

You then compare each record signature with the composite key signature. Only the record signature of the string "Jill walks up the lane" matches (for every b_i equal to 1 in the composite key signature, a b_i equals 1 in the record signature).

The program identifies the string even though the search key "walk" is only a substring of the word "walks". This is an important feature of the signature screening method. You need to specify only three or more letters of a word to find it. The substring can be from any position in the word.

Signature Screening on an 8-Bit Machine

The detective who helped us earlier has gone home to watch reruns of Perry Mason, and it's time to get to work. First, I'll describe the general aspects of implementing signature screening on 8-bit machines. I'll follow this with an explanation and complete Assembly-language listings of two signature screening programs (Mindex and Search) for the Models II, 12, 16, III, and 4.

Finally I'll give instructions on using the programs, suggest potential applications, and venture a prediction on how personal computers will store and retrieve textual information in the future.

Source and Index File Structure

Start with a simple case in which each text record is 255 or fewer bytes and is followed by an end-of-record indicator (ASCII 13). The program stores only one record per disk sector. Figure 5 shows a 10-sector source file, the records in each sector, and the corresponding index file.

The index file has the same name and password as the source file, but the extension MAP. The first sector of the index file, called the header, contains the name and version of the indexing program (Mindex 1.2) and the source file's file specification. The source file's drive number isn't included since the user might wish to switch drives.

The remainder of the index file sectors are called signature sectors. Each signature sector contains eight record signatures stored in parallel. Think of it as eight 251-bit strings stacked one on top of another. The record signature of the n^{th} source file record is located in the $(INT(n/8)+1)^{th}$ sector of the index file and comprises the $(n \bmod 8)^{th}$ bit of bytes zero to 250.

Neither Mindex nor Search uses the 5 bytes (251 to 255) at the end of each signature sector. In Fig. 5, you can find the signature of the first source record (sector zero) by reading the first bit (bit

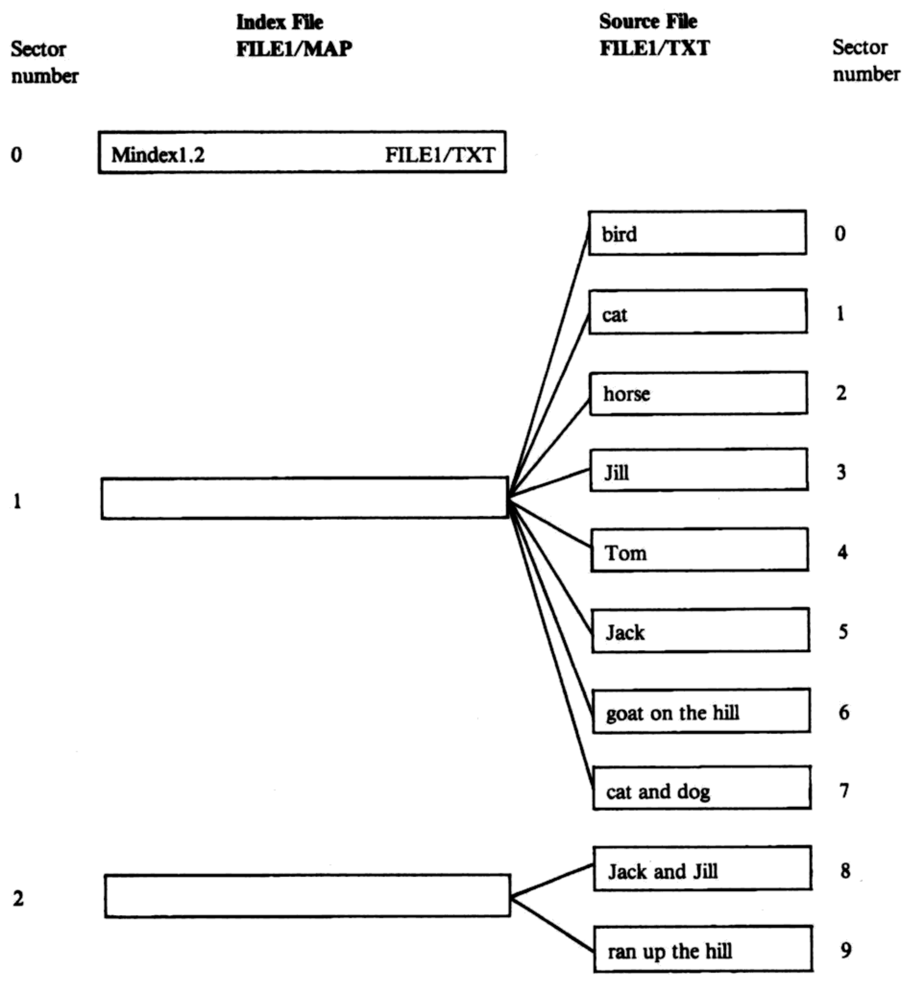


Figure 5. Source file (FILE1/TXT) and its index file (FILE1/MAP).

```
                                        Signature                                                                String
8         9         1         1         1         1         1         1         1         1         1         1         2         2
0         0         0         1         2         3         4         5         6         7         8         9         0         1
                    0         0         0         0         0         0         0         0         0         0         0         0

0001000000000000000001000000100000000000000000000000000000000000000000000000000000000000000000000000010001000001000000000000000000000  He ran down the lane
0001000000000010000000000000001000000000000000000000000000000000001000000000000000000000000000000000100100000000001001000000000000000000001  Jill walks up the lane
0001000000000000000000000000001000000000000000000000000000001000000000000000000000000000000000000000000100000000001000000000000000000001  The goat ran up the lane

00000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000000000000000000000001  Jill
0001000000000010000000000000001000000000000000000000000000000000000000000000000000000000000000010000000000100000000000000000000000000  walks + lane
```

Figure 4. Record signatures of the three strings in Table 2 and key signatures of the search keys "Jill" and "walks + lane".

EXTRAORDINARY VALUES

HARD DISK

5 MEG FIXED + 5 MEG REMOVABLE	$2895.00
10 MEG FIXED	2295.00
10+10 MEG FIXED	3495.00
10 MEG FIXED + 5 MEG REMOVABLE	3395.00
15 MEG FIXED	2395.00
15+15 MEG FIXED	3795.00
15 MEG FIXED + 5 MEG REMOVABLE	3495.00

ADAPTOR MODULES for:
TRS-80 Mod I, III, 4, LNW-80, -II,
IBM-PC, -XT, XEROX 820, -II
Coming soon adaptors for Apple-II, IIe, Franklin

Under desk mounting bracket	34.95
Boot ROM for Model III, 4	34.95
Extended Warranty (to 1 year)	150.00

MODEL 4

$1649.00

128K Model 4

2 Drives, VR-R-S232C

16K to 64K RAM upgrade kit	$96.00
64K to 128K RAM upgrade (includes PAL)	110.00
PAL only for 64K to 128K upgrade	34.95

DISK III

Internal floppy disk subsystem for Model III & 4 (includes controller, power supply, cabling, bracket and all associated mounting hardware).

DISK III KIT — Without drive	$299.95
DISK III KIT — With 1 drive (ss, dd)	449.00
DISK III KIT — With 2 drives (ss, dd)	699.00

EXTERNAL FLOPPY DISK DRIVES (ss,dd)		$229.00
80 Track or 2 sides	add, per dr	100.00
80 Track and 2 sides	add, per dr	200.00
EXTERNAL Power supply & enclosure only		54.95
VR-RS232C		80.00

Model III & 4 — Direct replacement for 26-1148

SPECIAL BUYS ON COD & PREPAID ORDERS ONLY:

EPSON FX80, MX80, MX100	CALL
C ITOH F10	1395.00
C ITOH TRACTOR	225.00
MX80 RIBBON	9.95
MX100 RIBBON	19.95
C ITOH & DIABLO RIBBONS	7.50
EPSON GRAPHTRAX	75.00
PAPER 9½ x 11 FANFOLD	29.95
PAPER 14⅞ x 11 GREENBAR	34.95
PRINTER CABLES	25.00
OKIDATA MICROLINE 82A	429.00
OKIDATA MICROLINE 83A	699.00
PERCOM DOUBLER II	169.00
PERCOM DATA SEPARATOR	29.95
LNW RESEARCH (FULL LINE)	CALL
DOS-PLUS OPERATING SYSTEM	149.95
VERBATIM 525-01	26.90
DC HAYES SMARTMODEM 300	229.00
DC HAYES SMARTMODEM 1200	569.00

Call Toll Free • 800-345-8102

Published prices reflect cash discount. All prices are subject to change without notice. TRS-80 and TRSDOS are trademarks of Tandy Corp.

Telephone Hours: 8:30am—7pm MON-FRI., SAT 10-3 EST
CABLE "VRDATA" TELEX 845-124

MasterCard VISA® AMERICAN EXPRESS®

VR data ✓10

(215) 461-5300
777 Henderson Blvd.,
Folcroft, PA 19032

VR DATA — WEST
WATONGA, OK
1-405-623-8664

✓ See List of Advertisers on Page 291

zero) of each of the bytes in the first signature sector.

Indexing

I'll index the file in Fig. 5. First, the program reserves a 256-byte buffer in memory for the signature sector. After it opens the source and index files and writes the index file header sector, the program starts indexing.

The program sets all the bytes in the signature sector buffer to zero. The program reads the first source file record into memory. It then hashes each triplet in the source record to a number between zero and 250, and sets bit zero of the corresponding byte in the signature sector buffer to 1. In the first source record the triplets "bir" and "ird" hash to 171 and 194, respectively, so the program sets the first bits of bytes 171 and 194 of the signature sector to 1.

After the program processes the first record, it reads the second record (sector 1) into memory. Again, the program hashes each triplet in the record, but this time the program sets bit 1 of the corresponding byte equal to 1. The triplet "cat" hashes to 193 so the program sets bit 1 of byte 193 of the signature sector to 1.

The program repeats this procedure until it indexes eight of the records. It then stores the signature sector buffer in the index file (sector 1). The program processes the next eight sectors of the source file in the same manner, and sequentially stores the signature sector in the index file. Indexing continues until the end of the source file, then the program writes the last signature sector to the index file and closes both files.

Searching

Remember that in its simplest form the search algorithm is: Match the signature of the first record to the key signature. If, for every b_i equal to 1 in the key signature, you have a corresponding b_i equal to 1 in the record signature, the screening test is positive and the program brings the record into memory and searches it for the key. Continue screening record signatures until you reach the end of the index file.

An ideal way to compare the key signature to a record signature is to have a machine with 251-bit registers and accumulators. You could compare the key and record signatures in one operation. It would be a form of parallel processing—the same as putting multiple needles in edge-notched cards and selecting only those cards that fall out.

Search key	Jill	
Triplets	jil	ill
Hash values	212	142
Key hash buffer	21 14 22	

Figure 6. Search key "Jill" for search of the source file in Fig. 5.

Eight-bit machines, however, necessitate the use of serial processing. Although a program can sequentially compare all 251 bits of the key and record signatures, it would be wasteful because the majority of bits in the key signature are set to zero and can be ignored.

Instead, the program stores the hash values of all triplets in the search key in a separate buffer called the key hash buffer (Fig. 6). If, for each hash value in the key hash buffer, the corresponding

Listing 3 continued

```
;----------------------------------------------------------------
                                ;MODII buffers
                                ;-------------------------------
        PUBLIC  ACCESS          ;access mode of OPEN routine
        PUBLIC  BREAK           ;break endabled buffer (Y/N)
        PUBLIC  INDCB           ;INput (source) file DCB
        PUBLIC  IDCBRL          ;Input file DCB Record Length
        PUBLIC  IDCBCR          ;Input file DCB Current Record
        PUBLIC  IDCBLR          ;Input file DCB Last Record
        PUBLIC  INLLR           ;true last record of source file
        PUBLIC  IDCBOS          ;Input file DCB xxxxxxx
        PUBLIC  MAPDCB          ;MAP (index) file DCB
        PUBLIC  MDCBRL          ;Map file DCB Record Length
        PUBLIC  MDCBCR          ;Map file DCB Current Record
        PUBLIC  MDCBLR          ;Map file DCB Last Record
        PUBLIC  MAPLLR          ;true last record of index file
        PUBLIC  MDCBOS          ;Map file DCB xxxxxxxx
        PUBLIC  MEMEND          ;contents ->top of memory
        PUBLIC  OUTCHR          ;record buffer (1 byte) for output file
        PUBLIC  OUTDCB          ;output file DCB
        PUBLIC  PRTBUF          ;printer buffer (80 char)
        PUBLIC  SORLN           ;start of record line no.
        PUBLIC  STMAP           ;start of MAP index file (always last byte
                                ;of SEARCH program)
;----------------------------------------------------------------
                                ;MODII I/O parm
                                ;-------------------------------
        PUBLIC  ARROWD          ;arrow down
        PUBLIC  ARROWL          ;arrow left
        PUBLIC  ARROWR          ;arrow right
        PUBLIC  ARROWU          ;arrow up
        PUBLIC  BRKKEY          ;break key
        PUBLIC  BSPKEY          ;back space key
        PUBLIC  CPYPOS          ;copyright msg position
        PUBLIC  DELKEY          ;delete key
        PUBLIC  DFTFS           ;default file size
        PUBLIC  INSKEY          ;insert key
        PUBLIC  LSBYTE          ;col of last byte in 255 byte rec
        PUBLIC  PENULN          ;penultimate line (next to bottom)
        PUBLIC  SWIDTH          ;screen width
        PUBLIC  SWM1            ;screen width minus 1
        PUBLIC  TOPLN           ;top line of editor screen
;----------------------------------------------------------------
                                ;external routines and buffers
                                ;referenced by MODII
                                ;-------------------------------
        EXTERN  ANSWER          ;general purpose 1 byte buffer
        EXTERN  EATBNK          ;bypass leading blanks
        EXTERN  EATNBNK         ;bypass leading non-blank char
;----------------------------------------------------------------
                                ;Model II equates
                                ;-------------------------------
ARROWD  EQU     1FH             ;arrow down
ARROWL  EQU     1CH             ;arrow left
ARROWR  EQU     1DH             ;arrow right
ARROWU  EQU     1EH             ;arrow up
BRKKEY  EQU     3               ;break key
SWIDTH  EQU     80              ;screen width
SWM1    EQU     79              ;screen width minus 1
TOPLN   EQU     19              ;top line of edit text
PENULN  EQU     22              ;penultimate line
LSBYTE  EQU     14              ;column of last byte in 255 byte rec
DFTFS   EQU     800             ;default file size for 3by5
DELKEY  EQU     2               ;delete key (F2)
INSKEY  EQU     1               ;insert key (F1)
BSPKEY  EQU     8               ;backspace key
CPYPOS  EQU     52              ;copyright notice position
;----------------------------------------------------------------
                                ;HOME
                                ;-------------------------------
                                ;pos cursor to r1:c1
                                ;reg altered: AF,D
                                ;-------------------------------
                                ;BOTTOM
                                ;-------------------------------
                                ;pos cursor to r24:c1
                                ;reg altered: AF,D
HOME    PUSH    BC
        LD      BC,0            ;r1:c1
HOME5   LD      D,0
        SVC     10              ;pos cursor
        POP     BC
        RET
;
BOTTOM  PUSH    BC              ;pos cursor to r24:c1
        LD      BC,23*256+0     ;r24:c1
        JR      HOME5
;----------------------------------------------------------------
                                ;NEWLN
                                ;-------------------------------
                                ;rolls screen and positions
                                ;cursor to next to last line
                                ;also keeps track of first line of
                                ;current (or last) record displayed
                                ;reg altered: A
NEWLN   PUSH    BC
        PUSH    DE
        PUSH    HL
        CALL    BOTTOM          ;pos to bottom line
        LD      A,0DH
        CALL    VDCHAR          ;roll screen by dsp EOR on bottom line
        LD      BC,22*256+0     ;r23:c1
        CALL    POSCUR
        LD      A,(SORLN)
        CP      4               ;has first line reached top of screen?
        JR      Z,NEWLN5        ;if it has do not dec (SORLN)
        LD      A,(SORLN)
        DEC     A               ;dec (SORLN)
        LD      (SORLN),A
NEWLN5  POP     HL
        POP     DE
        POP     BC
        RET
;----------------------------------------------------------------
                                ;CURPOS
                                ;-------------------------------
                                ; get cursor position
                                ;value returned in BC
                                ;B=row, C=column
                                ;reg altered: AF,D
CURPOS  LD      D,0
        SVC     11              ;get cursor pos
        RET
;----------------------------------------------------------------
                                ;POSCUR
                                ;-------------------------------
                                ;position cursor
                                ;B=row, C=column
                                ;reg altered: AF,D
POSCUR  LD      D,0
        SVC     10              ;pos cursor
        RET
;----------------------------------------------------------------
                                ;KBINIT
                                ;-------------------------------
KBINIT  SVC     1               ;initilize keyboard
        RET
;----------------------------------------------------------------
                                ;KBCHAR
                                ;-------------------------------
                                ;get char from keyboard
                                ;works like Model III KBCHAR
                                ;value of pressed key returned
                                ;in A reg.  If no key pressed, A=0
KBCHAR  PUSH    BC
        LD      B,0
        LD      A,'N'
        LD      (BREAK),A       ;set break switch to N (no break)
        SVC     4               ;Model II KBCHAR
        LD      A,(BREAK)
        CP      'Y'             ;was break key pressed
        JR      Z,KBCH6
        LD      A,B             ;move entered char to A reg
        POP     BC
        RET
```

Listing 3 continued

✓ See List of Advertisers on Page 291

record signature bit equals 1, the screening test is positive, and the program searches the source record for the key. Since it isn't efficient to compare 1 bit at a time on an 8-bit machine, the program processes eight record signatures at once.

To search the file in Fig. 5, you must first open the index file, then the source file specified in the index file header. The search begins when you enter a key (e.g., the name "Jill").

The program hashes each triplet of the key and stores the hash value sequentially in the key hash buffer (Fig. 6). In the example, it stores the hash values 212 (h(jil)) and 142 (h(ill)) respectively in the first and second bytes of the key hash buffer.

Then the program reads the first signature sector of the index file into the signature sector buffer in memory. It sets the accumulator (A) to 255 (1111 1111 binary). The program ANDs byte 212 (hash value of the triplet "jil") of the signature sector with A.

If A equals zero, the triplet "jil" isn't present in the first eight sectors of text and you can process the next signature sector. If A doesn't equal zero, the triplet "jil" might be present, so the program continues to process the current signature sector.

The program ANDs byte 142 (hash value of the triplet "ill") of the signature sector with A. Skip to the next signature sector if A equals zero; otherwise continue as follows. For each bit in A equal to 1 it's possible (but not certain) that the corresponding record contains the triplets "jil" and "ill".

In the example, A equals 72 (0100 1000 binary), indicating that sectors 3 and 6 might contain the key you're looking for. The program reads record 3 into memory and searches sequentially for the string "Jill". It finds the key and displays the record.

Then the program reads record 6 into memory, but the key "Jill" isn't present so it bypasses the record. The reason for the collision is simple—the triplet "goa" hashes to the same value as "jil" and the triplet "ill" appears in both records.

The program sequentially reads the index file a sector at a time and processes each signature sector similarly. After it screens the entire index file and searches the appropriate source records, you can enter another key to restart the searching process.

Searching for the joint occurrence of two or more keys is merely an extension of the above procedure (search the file for the names "Jack" and "Jill"). The program stores the hash values (123, 4, 212, and 142) of the triplets "jac", "ack", "jil", and "ill" in the key hash buffer.

"The program processes eight record signatures at once."

The program reads the first signature sector into memory and sets the accumulator to 255. It ANDs byte 123 of the signature sector with A. Similarly, it ANDs bytes 4, 212, and 142 of the signature sector with A.

If at any time A equals zero, none of the eight source records contains both strings, and the search skips to the next signature sector. In the example, only bit 5 equals 1, so it's the only source record searched. The program finds the names "Jack" and "Jill" and displays the record.

In these simple examples only one record is present per sector. When two or more records completely fit into one sector, the program indexes the entire sector as one record. When the program searches the sector, however, it examines each record separately for the search key(s). If a record spans a sector boundary (starts in one sector and ends in another), it is considered as logically in the first sector in which it begins.

You increase the speed of searching by storing as many signature sectors as possible in memory (reducing the number of disk reads during index screening).

TRS-80 Signature Screening

Mindex and Search are Assembly-language programs that use signature screening to search text files. Mindex indexes the source file and stores the index on disk. Search uses the index created by Mindex to search large text files quickly.

These programs were written with the Radio Shack Assembly Language Development System (ALDS) on a Model II. Each program consists of several modules. The two input/output (I/O) modules include MODII/SRC for the Model II and MODIII/SRC for the Model III.

The ALDS editor, assembler, and linker work only on the Model II. If you have a Model III and want to use these programs exactly as written, you must first write, assemble, and link them on a Model II and then, using the ALTRAN

Listing 3 continued

```
KBCH6   LD      A,3             ;<break> pressed so A=3
        POP     BC
        RET
;-------------------------------------------------------------
                                ;KBWAIT
                                ;------------------------------
                                ;wait for char from keyboard
                                ;works like Model II KBWAIT
KBWAIT  PUSH    BC
        LD      B,0
        LD      A,'N'
        LD      (BREAK),A       ;initialize to no break
KBW4    SVC     4
        PUSH    AF
        LD      A,(BREAK)
        CP      'Y'             ;<break> pressed?
        JR      Z,KBW6          ;jp to break porcessing section
        POP     AF
        JR      NZ,KBW4         ;loop until key pressed
        LD      A,B             ;move entered key to A reg
        POP     BC
        RET
KBW6    POP     AF
        LD      A,3             ;break key entered
        POP     BC
        RET
;-------------------------------------------------------------
                                ;WPKBWT
                                ;------------------------------
                                ;word processing KBWAIT routine
                                ;on the Model II this is basically
                                ;the same as the KBWAIT routine
WPKBWT  PUSH    BC
        LD      B,1             ;blinking cursor on
        SVC     8
        LD      B,0
WPKBW4  CALL    KBWAIT
        LD      C,A             ;save keyed in char
        LD      B,2             ;cursor off
        SVC     8
        LD      A,C             ;restore keyed in char
        POP     BC
        RET
;-------------------------------------------------------------
                                ;OPEN
                                ;------------------------------
                                ;open file
                                ;entry: HL-> buffer
                                ;       DE->DCB
                                ;exit:  A=TRSDOS error msg
                                ;reg altered: AF,HL
OPEN    LD      (BUFADR),HL
        LD      HL,BUFADR
        SVC     40              ;open file SVC
        PUSH    AF
        LD      A,0
        LD      (RL),A          ;set record length to 256 bytes!
        POP     AF
        RET
;
BUFADR  DEFW    0               ;buffer address (passed by HL)
RECADR  DEFW    0               ;record address (not needed for 256 byte rec)
EODAD   DEFW    0
ACCESS  DEFB    'R'             ;read only access
RL      DEFB    0               ;record length (set to 0)
VORF    DEFB    'F'             ;variable or fixed len record
CREATE  DEFB    0
ATTRIB  DEFB    0
;-------------------------------------------------------------
                                ;INIT
                                ;------------------------------
                                ;initialize (create) a file
                                ;entry: HL->record buffer
                                ;       DE->DCB
                                ;exit:  A=TRSDOS error msg
                                ;reg altered: AF,HL
INIT    LD      (BUFADRI),HL
        LD      HL,BUFADRI
        SVC     40
        PUSH    AF
        LD      A,0
        LD      (RL),A          ;record lenght = 256
        POP     AF
        RET
;
BUFADRI DEFW    0               ;address of sector buffer
RECADRI DEFW    0               ;address of record buffer
EODADI  DEFW    0
ACCESSI DEFB    'W'             ;access = (W)rite
RLI     DEFB    0               ;record length = 256 bytes
VORFI   DEFB    'F'             ;fixed length records
CREATEI DEFB    2
ATTRIBI DEFB    0
;-------------------------------------------------------------
                                ;OPINIT
                                ;------------------------------
                                ;open a file but do not reset it
                                ;entry: HL-> sector read buffer
                                ;       DE->DCB
                                ;exit:  A=TRSDOS error msg
                                ;reg altered: AF,DE,HL
OPINIT  LD      A,0
        LD      (CREATEO),A     ;open only do not reset
        LD      BC,OUTCHR       ;record buffer
        LD      (RECADRO),BC
        LD      (BUFADRO),HL    ;sector buffer address
        LD      HL,BUFADRO
        PUSH    DE              ;save DE & HL in case open fails
        PUSH    HL
        LD      A,(ANSWER)
        CP      'R'
        JR      Z,OPIN3
        SVC     40              ;open
        JP      Z,OPIN4
OPIN3   LD      A,2             ;create and reset
        LD      (CREATEO),A
        POP     HL
        POP     DE
        SVC     40              ;open
        RET
OPIN4   POP     HL              ;restore stack
        POP     DE
        RET
;
BUFADRO DEFW    0
RECADRO DEFW    0
EODADO  DEFW    0
ACCESSO DEFB    'W'
RLO     DEFB    1
VORFO   DEFB    'F'
CREATEO DEFB    0
ATTRIBO DEFB    0
OUTCHR  DEFB    0               ;record buffer
;-------------------------------------------------------------
                                ;CLOSE
                                ;------------------------------
                                ;close file
                                ;entry: DE->DCB
                                ;exit:  A=TRSDOS error msg
CLOSE   SVC     42              ;close file
        RET
;-------------------------------------------------------------
                                ;SETMLR
                                ;------------------------------
                                ;set MAP last record to true last rec
                                ;this routine is necessary to make programs
                                ;compatible with Model III
SETMLR  LD      HL,(MDCBLR)
        LD      (MAPLLR),HL     ;no need to dec HL on Mod II
        RET
```

Listing 3 continued

See List of Advertisers on Page 291

programs supplied in the ALDS package, transfer them to the Model III.

Alternatively, you can write and assemble the programs using a simple editor/assembler such as Radio Shack's Series I Editor/Assembler (EDTASM). EDTASM does not contain a linker so it's necessary to combine the modules into one program for Mindex and Search.

I first developed Mindex and Search (version 1.1) on a Model III using EDTASM. However, the editor's memory was limited and the program comments were restricted.

When I switched to the Model II, I started using ALDS. Much to my chagrin, the operating systems of the two machines are quite different. For example, to display a character on the Model III, you load the character into the accumulator and make a call to location 33 hexadecimal (hex). The routine alters the contents of the DE register pair.

On the Model II, you load the character into register B, load the supervisor call code 8 into the accumulator, and execute an RST 8 operation. The DE register pair isn't altered.

Rather than rewrite the program for the Model II, I developed separate I/O routines for the Models II and III. Since I wrote the original programs for the Model III, the parameters pass to the I/O routines in Model III format.

For example, the program stores the character you want displayed on the screen in the accumulator. A few of the Model II routines (SVCs) aren't present in Model III TRSDOS so I added them (such as routines to position the cursor at coordinates stored in the BC register pair).

If you're using Radio Shack's ALDS, type in the program modules in Program Listings 1–3 and 7–12 for the Model II, and Listings 4–12 for the Model III. After the ALDS writes each module, you must assemble them to relocatable code by using ALASM, the ALDS assembler.

After ALASM assembles all modules, you're ready to link them. For the Model II, type ALLINK MINDEX2 MINDEX $=4000 to link the Mindex program and ALLINK SEARCH2 SEARCH $=4000 to link the Search

Listing 3 continued

```
;-------------------------------------------------------------
                                ;SETILR
                                ;-----------------------------
                                ;set input last rec to true last rec
SETILR  LD      HL,(IDCBLR)
        LD      (INLLR),HL      ;true value regardless of whether
                                ;sector is full or not!!
        RET
;-------------------------------------------------------------
                                ;POSN
                                ;-----------------------------
POSN    RET                     ;not used for Model II
                                ;but present for Mod III compatibility
;-------------------------------------------------------------
                                ;POSEOF
                                ;-----------------------------
                                ;position to end of file
                                ;entry: DE->DCB
POSEOF  LD      BC,0FFFFH
        SVC     35
;
        LD      DE,OUTDCB
        SVC     33              ;LOCATE
        DEC     BC              ;must have this!!
        LD      DE,OUTDCB
        SVC     35              ;sets file cursor to last char
;
        RET
;-------------------------------------------------------------
                                ;READDR
                                ;-----------------------------
READDR  SVC     35              ;read direct
        RET                     ;entry: DE->DCB,BC=rec no.
;-------------------------------------------------------------
                                ;READNX
                                ;-----------------------------
READNX  SVC     34              ;read next record
        RET                     ;entry: DE->DCB
;-------------------------------------------------------------
                                ;WRITE
                                ;-----------------------------
WRITE   SVC     44              ;write direct
        RET                     ;DE->DCB, BC=rec no.
;-------------------------------------------------------------
                                ;WRITNX
                                ;-----------------------------
WRITNX  SVC     43              ;write next record
        RET                     ;entry: DE->DCB
;-------------------------------------------------------------
                                ;KBLINE and KBLIN3
                                ;-----------------------------
                                ;KBLINE: EOR=0DH, KBLIN3: EOR=3
                                ;entry: HL->buf
                                ;B = max length
                                ;EOR=0DH or 03
                                ;returns with last char entered in A
KBLIN3  LD      A,3             ;EOR=3
        JR      KBL03
KBLINE  LD      A,0DH
KBL03   LD      (EORCHR),A
        LD      A,1
        CALL    VDCHAR          ;blinking cursor
        LD      C,0
KBL05   CALL    KBWAIT
        LD      D,A
        CP      0DH             ;<enter> key?
        JR      Z,KBL0DH
        CP      08              ;<backspace> key?
        JR      Z,KBLBSP
        CP      BRKKEY          ;<break> key?
        JR      Z,KBBRK
        CP      20H
        JP      M,KBL05
        LD      A,B
        CP      C
        JR      Z,KBL05
        LD      A,D             ;reload since A used for calc
        LD      (HL),A          ;store in buf
        INC     C
        INC     HL
        CALL    VDCHAR
        JR      KBL05
KBLBSP  LD      A,0             ;back space routine
        CP      C
        JR      Z,KBL05         ;do nothing if start of buf
        LD      A,8
        CALL    VDCHAR          ;dps backspace
        DEC     HL
        DEC     C
        JR      KBL05
;
KBBRK   LD      A,(EORCHR)      ;break routine
        LD      (HL),A
        CALL    VDCHAR          ;display EOR
        LD      A,BRKKEY        ;return with A=break
        LD      (EORCHR),A      ;save for ret
        JR      KBL0DI
KBL0DH  LD      A,(EORCHR)      ;place EOR (03 or 0DH)
        LD      (HL),A
        CALL    VDCHAR          ;display EOR
KBL0DI  INC     HL              ;HL->byte past EOR
        LD      A,2
        CALL    VDCHAR          ;cursor off
        LD      A,(EORCHR)      ;A returns with last char (0DH or 03)
        LD      B,C             ;char count
        RET
EORCHR  DEFB    0DH             ;EOR character
;-------------------------------------------------------------
                                ;PRINIT
                                ;-----------------------------
PRINIT  PUSH    BC              ;initialize printer
        PUSH    DE
        SVC     17              ;printer init
        POP     DE
        POP     BC
        RET
;-------------------------------------------------------------
SCRPRO  PUSH    BC              ;scroll protect
        LD      B,A             ;A contains line no.
        SVC     27
        POP     BC
        RET
;-------------------------------------------------------------
                                ;ERASES
                                ;-----------------------------
ERASES  LD      A,18H           ;erase to end of screen code
        CALL    VDCHAR
        RET
;-------------------------------------------------------------
ERASEL  LD      A,17H           ;erase to end of line
        CALL    VDCHAR
        RET
;-------------------------------------------------------------
                                ;VDCHAR
                                ;-----------------------------
                                ;display char
VDCHAR  PUSH    BC
        PUSH    AF
        LD      B,A
        SVC     8
        POP     AF
        POP     BC
        RET
;-------------------------------------------------------------
                                ;VDINIT
                                ;-----------------------------
VDINIT  PUSH    BC              ;clear screen and initialize
        LD      BC,257
        SVC     7
        POP     BC
        RET
;-------------------------------------------------------------
                                ;VDLINE
                                ;-----------------------------
                                ;entry: HL->string
                                ;0DH or 03 stop display
                                ;exit: HL ->next char
VDLINE  PUSH    BC
VDL05   LD      B,(HL)
        SVC     8               ;vdchar
        INC     HL
        LD      A,B
        CP      0DH
        JP      Z,VDL12
        CP      03
        JP      Z,VDL12
        JP      VDL05
VDL12   LD      A,2             ;cursor off
        CALL    VDCHAR
        POP     BC
        RET
;-------------------------------------------------------------
                                ;VDLIN3
                                ;-----------------------------
                                ;display line but no carriage return
                                ;entry: HL->string
                                ;0DH or 03 stop display
                                ;DOES NOT PRINT 0DH!!
                                ;exit: HL ->next char
VDLIN3  PUSH    BC
VDL305  LD      B,(HL)
        LD      A,0DH
        CP      B
        JR      Z,VDL308
        SVC     8               ;vdchar
VDL308  INC     HL
        LD      A,B
        CP      0DH
        JR      Z,VDL312
        CP      03
        JR      Z,VDL312
        JR      VDL305
VDL312  LD      A,2             ;cursor off
        CALL    VDCHAR
        POP     BC
        RET
;-------------------------------------------------------------
                                ;VDCLS
                                ;-----------------------------
VDCLS   LD      B,1BH           ;clear screen below cursor code
        SVC     8               ;VDCHAR
        RET
;-------------------------------------------------------------
                                ;KEYIN
                                ;-----------------------------
```

Listing 3 continued

PUT PRICES IN CHECK

RADIO SHACK®*
ZIP BOX RELOADS FOR
LINE PRINTER I, II, & IV
$2.77 EA. $29.92 DOZ

NEW ITEM
CARTRIDGE RIBBONS FOR
RADIO SHACK®*
LINE PRINTER VI & VIII
$7.99 EA $86.29 DOZ

CARTRIDGE RIBBONS FOR
RADIO SHACK®*
LINE PRINTER III & V
NEWLY DESIGNED CARTRIDGE
$6.49 EA. $70.09 DOZ

INNOVATIVE CONCEPTS
FLIP'N'FILE
DISC STORAGE BOX
HOLDS UP TO 60 DISKETTES

5¼″	8″
$24.95 EA	$29.95 EA

CARTRIDGE RIBBONS FOR
RADIO SHACK®*
DAISY WHEEL & DAISY WHEEL II
MULTI-STRIKE & NYLON
$5.99 EA $64.69 DOZ

PROTECTALL LINE VOLTAGE
SURGE SUPPRESSOR
(SIX OUTLET)
$39.95 EA

CARTRIDGE RIBBONS FOR
APPLE PRINTERS
NEC 8023A
C. ITOH PROWRITER
$9.95 EA $107.46 DOZ

MEMOREX DISKETTES
5¼ SINGLE-SIDE DUAL DENSITY
#3481
$24.99 10 PACK

MAXELL DISKETTES
5¼″ SINGLE SIDE
DUAL DENSITY
MD-1
$29.90 10 PACK

CARTRIDGE RIBBONS FOR
EPSON

MX-80	MX-100
$6.99 EA.	$11.95 EA.
$75.49 DOZ.	$129.06 DOZ.

RIBBONS FOR
IDS PRINTERS

	EA	DOZ
440	$2.75	$29.70
PAPER TIGER	6.95	75.06
MICROPRISM	7.99	86.29
PRISM	10.95	118.25

LABEL SPECIAL
$2.99 /K
(5K MIN)
1 ACROSS 3½ x 15/16 CONTINUOUS LABELS

MOST RIBBONS AVAILABLE IN COLORS TOO!
CALL OR WRITE FOR OUR SUPPLIES CATALOGUE
ON ORDERS UNDER $14.00 PLEASE ADD $3.00 FOR SHIPPING
MINIMUM RIBBON ORDER $30.00 OR 1 DOZEN

MasterCard

Check-Mate™

✓466

VISA

TOLL FREE 800-343-7706
IN MASS 617-963-7694
PHONES OPEN 9AM-6PM EASTERN TIME

51 DIAUTO DR. P.O. BOX 103
RANDOLPH, MA 02368

MASS RESIDENTS
ADD 5% SALES TAX

*RADIO SHACK IS A REGISTERED TRADEMARK OF THE TANDY CORP.

```
MAP INDEX PROGRAM Version 1.2                    © 1983 Softshell

File Name:

Enter name of file to be indexed, or ? for help
```

Figure 7. The Mindex screen.

program. Do the same for the Model III, but use the file names MINDEX3 and SEARCH3 and set $=5200 (the program's starting location).

To transfer the programs from the Model II to a Model III, you must use the ALTRAN transfer programs supplied with ALDS. On the Model III, Mindex and Search must have the extension CMD; on the Model II, no extension is necessary.

Using Mindex and Search

After you assemble and link both programs, you're ready to begin indexing and searching. The first requirement is an ASCII text file. You can create the text with Scripsit and convert it to ASCII file format by using the Scripsit convert utility.

If you don't have Scripsit or some other word processor/text editor, you can create text files by using the editor that comes with Basic. Instead of typing in a line of code, type in a record. You must save text files created with the Basic editor in ASCII format.

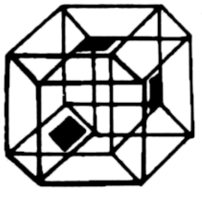
Mindex indexes text or data source files. It stores the index in the separate MAP index file. The program segments records longer than 255 characters into blocks of 255 characters and indexes them until it finds a block of characters less than 255 characters long terminating in an end-of-record (EOR) message. The index file that Mindex creates has the same name as your source file (unless you specify otherwise) and the extension MAP.

To run Mindex, type MINDEX and press the enter key. Mindex displays the screen in Fig. 7 and waits for you to enter the file specifications of the source and index files. The syntax is: SOURCE

Listing 3 continued

```
                                ;routine to input search keys
                                ;entry: HL -> input buffer
                                ;B = max no. of char
                                ;exit:  HL ->EOR
                                ;B=no. char, A= EOR
KEYIN   LD      A,1
        CALL    VDCHAR          ;turn cursor on
        LD      A,17
        CALL    VDCHAR          ;erase to end of line
        LD      C,0             ;init char count to 0
GETCHR  NOP
        CALL    KBWAIT
        CP      1               ;was F1 pressed?
        JP      Z,ENTAND        ;if so jp to '+' char entry
        CP      2               ;was F2 pressed?
        JP      Z,ENTOR         ;if so jp to '|' char entry
        CP      20H             ;char or control?
        JP      M,CTRL          ;control if less 20H
        JP      ALPHA           ;alphanumeric entered
ENTAND  LD      A,C             ;enter '+' char in search key
        CP      B
        JP      Z,GETCHR
        INC     C
        LD      A,'+'
        JP      ALPHA5
ENTOR   LD      A,C             ;enter '-' char in search key
        CP      B
        JP      Z,GETCHR
        INC     C
        LD      A,124
        JP      ALPHA5
CMDAND  LD      A,253           ;and operation code
        LD      (HL),A          ;store and cmd
        CALL    HON             ;reverse video on
        LD      A,'+'
        CALL    VDCHAR
        CALL    HOFF            ;reverse video off
        INC     HL
        JP      GETCHR
CMDOR   LD      A,254           ;or code
        LD      (HL),A
        CALL    HON
        LD      A,124
        CALL    VDCHAR
        CALL    HOFF
        INC     HL
        JP      GETCHR
;
                                ;alpha numeric char
ALPHA   LD      D,A             ;temp store A
        LD      A,C             ;A now = char count
        CP      B               ;cp to max char
        JP      Z,GETCHR        ;loop until CTR or CMD
                                ;char ok so process
        INC     C               ;bump counter
        LD      A,D             ;restore A
        CP      '+'
        JP      Z,CMDAND
        CP      124
        JP      Z,CMDOR
ALPHA5  LD      (HL),A
ALPHA7  INC     HL              ;bump HL
        CALL    VDCHAR          ;display char
        JP      GETCHR
CTRL    NOP                     ;control char entered
        CP      0DH             ;? CR
        JP      Z,CTRL0D
        CP      08              ;? backspace
        JP      Z,CTRL08
        CP      BRKKEY          ;break?
        JP      Z,CTRL01
        JP      GETCHR          ;other CTRL not allowed
CTRL01  NOP                     ;break key routine
        LD      A,C
        CP      0               ;was break hit on ct = 0?
        JP      NZ,CTRL02
        LD      A,02H
        CALL    VDCHAR          ;turn off cursor
        LD      A,BRKKEY
        LD      B,0
        RET
CTRL02  LD      A,BRKKEY
CTRL0D  NOP                     ;EOR
        LD      (HL),A
        JP      KBEND
CTRL08  NOP                     ;backspace routine
        LD      D,A             ;temp save A in D
        LD      A,C
        CP      0               ;if zero cannot backspace
        JP      Z,GETCHR
        DEC     C               ;dec counter
        DEC     HL              ;dec buffer pointer
        LD      A,08
        CALL    VDCHAR          ;erase one char
        JP      GETCHR
KBEND   LD      B,C
        PUSH    AF
        LD      A,2
        CALL    VDCHAR          ;turn off cursor
        POP     AF
        RET
;-----------------------------------------------------------------
                                ;HON
                                ;--------------------------------
HON     LD      A,1AH           ;reverse video
        CALL    VDCHAR
        RET
;-----------------------------------------------------------------
                                ;HOFF
                                ;--------------------------------
HOFF    LD      A,19H           ;reverse video off
        CALL    VDCHAR
        RET
;-----------------------------------------------------------------
                                ;PRSCN
                                ;--------------------------------
PRSCN   NOP                     ;print screen
        LD      BC,0
        LD      E,0
        PUSH    BC
PRSCNLP NOP
        LD      HL,PRTBUF
        LD      D,80
        POP     BC
        PUSH    BC
        SVC     11              ;move line to PRTBUF
        POP     BC
        INC     B
        PUSH    BC
        LD      HL,PRTBUF
        LD      B,80
        LD      C,0DH
        SVC     19              ;PRLINE
        INC     E
        LD      A,24
        CP      E
        JP      NZ,PRSCNLP
        POP     BC
        RET
;.......
PRTBUF  DEFS    80              ;printer buffer
        DEFB    0DH
;-----------------------------------------------------------------
                                ;MOVELN
                                ;--------------------------------
MOVELN  NOP                     ;move char from screen to buffer
                                ;HL->buffer,D=no. char
                                ;BC=cursor coordinates
        SVC     11
        RET
;-----------------------------------------------------------------
                                ;PRLINE
                                ;--------------------------------
                                ;print line
                                ;HL->buffer
                                ;B=no. char, C=ending char
PRLINE  SVC     19
        RET
;-----------------------------------------------------------------
                                ;PRTBLN
                                ;--------------------------------
                                ;print penultimate line
PRTBLN  PUSH    BC
        PUSH    DE
        PUSH    HL
        LD      B,PENULN
        LD      C,0
        LD      D,80
        LD      HL,PRTBUF
        CALL    MOVELN          ;move next to last line to PRTBUF
;
        LD      HL,PRTBUF
        LD      B,80
        LD      C,0DH
        CALL    PRLINE
```

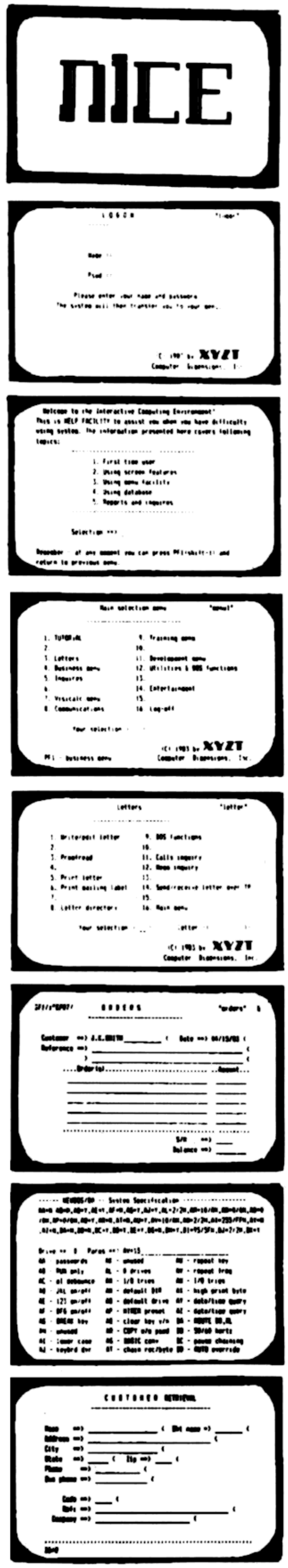
Listing 3 continued



After 2 years of extensive research and development XYZT Computer Dimensions, Inc is proud to announce ...

NICE™

the NEW INTERACTIVE COMPUTING ENVIRONMENT



NICE is nothing less than a comprehensive integration of all computing facilities – including DOS, numerous utilities, application programs, even games combined with menus, database, screens, libraries, reports – an interactive environment friendly to the first time user and the sophisticated systems intergrater alike. If you know what a turbo-charger can do for an auto engine, that is what NICE does for a computer. Clearly, it is the most exciting news to come along in software for the microcomputer

DO YOU KNOW HOW TO BOOT YOUR SYSTEM? THEN YOU KNOW HOW TO TURN YOUR COMPUTER INTO THE MOST POWERFUL USER FRIENDLY BUSINESS MACHINE!!

NICE makes development and using software – EASY. Based on a new concept this state-of-the-art software gives you the interactive power until now found only on mainframes, but at micro prices. Put a powerful menu-driven command structure at your fingertips! Run word-processors, spell-checkers, spread-sheet programs, utilities, application programs without remembering command sequence or constantly checking the manuals. Everything you need is on the screen. Create your own applications and add them to the system – use menus, relational files, customized screens, special forms. In a matter of hours, you zip through the programming projects you usually expected to take months. All thanks to a modular, consistently designed system.

IS YOUR WORDPROCESSOR WORKING? THEN THOSE SCREENS OUGHT TO BE IN YOUR SYSTEM !

Using a computer should not be difficult – no more complicated than driving a car. With NICE you are no longer required to know all of the intricate internal operations of the computer. Creation of all types of screens, menus, inquires and reports or customized forms is as simple as writing a letter – type it on your word-processor, save it to disk, and it is ready for use. You don't have to be a programmer – just a computer user.

It's NICE for you!

NICE system consists of:

Interactive Control & Programming Language (ICPL) .... $150.
Menu and Screen management facilities (MSF) .... $75.
Database facility (DBF) .... $75.
Forms and Report Writer (FRW) .... $75.
Library Support Option (LSO) .... $75.
Minimal NICE configuration — ICPL+MSF+LSO.

Minimal hardware requirements — 2 disk TRS-80® mod I/III. System is compatible with NEWDOS/80®-LDOS®-DOSPLUS®-MULTIDOS® floppy or hard disk.

Check, M.O.. C.O.D., VISA,
MC plus $4.00 s/h.
Foreign orders — extra $10.00

XYZT Computer Dimensions, Inc. ✓158
2 Penn Plaza, Suite 1500
New York, N.Y. 10121
(212) 244-3100
Order by Mail or Check Your Local Dealer

VISA

DEALER INQUIRIES WELCOME!

```
MAP SEARCH PROGRAM Version 1.2                    © 1983 Softshell

File Name:




Enter file names, or ? for help
```

Figure 8. Initial Search screen.

FILE,INDEX FILE;OPTIONS.

You can also enter the file specifications and options on the TRSDOS command line (MINDEX REF:0,:1;P). Only the source file specification is necessary; the program assigns the index file the same name, password, and disk drive as the source file unless you specify otherwise.

For example, typing REF/TXT.SECRET:1 and pressing the enter key opens the source file "REF/TXT.SECRET:1" and creates the index file "REF/MAP.SECRET:1", while typing REF/TXT.SECRET:1,.CODEWORD:2 and pressing the enter key opens the source file "FILE1/TXT.SECRET:1" and creates the index file "FILE1/MAP.CODEWORD:2".

In this case the index file has a different password from the source file and is on drive 2 rather than drive 1. You don't need to specify the disk drive for your source file unless you have two files with the same specification on two separate drives. The program places the index file on the same disk as the source file unless you specify another drive.

Indicate the paragraph format option by appending a semicolon and the letter P to the file specifications. In the paragraph format, the program treats each sentence as a string. You should use the paragraph format for text files composed of sentences arranged in paragraphs and that don't have an EOR at the end of each line or sentence.

MAP uses the period as a pseudo EOR and treats each sentence as a record. If you don't specify the indexing format, the program defaults to the record format. Typing TEXTFILE,:1;P and pressing the enter key opens the source file TEXTFILE from whatever drive it's on, creates the index file TEXTFILE/MAP on drive 1, and indexes the source file in paragraph format.

Mindex displays the name, extension, password, and drive of the source and data files, indicates that it's opened each file successfully (or gives a self-explanatory error message), and begins indexing. The program displays the number of the source file sector it indexes.

When the program finishes indexing, it saves the index file on disk and returns you to TRSDOS. You don't need to re-index the source file unless you modify the file or you delete the index file from the disk.

To run Search, type SEARCH and press the enter key. The screen in Fig. 8 appears. Type in the name of the file you want to search. Alternatively, you can enter the file specifications on the TRSDOS command line (SEARCH REF:1). The complete syntax is SOURCE FILE, INDEX FILE, OUTPUT FILE.

It's only necessary to type in the name of the source file. The program assigns the same specification to the index file (except for the extension MAP) and output file (except for the extension OUT) unless you specify otherwise.

For example, typing TESTFILE/TXT assigns the specification TESTFILE/MAP to the index file and TESTFILE/OUT to the output file. The MAP index file contains the source file name, extension, and password so you don't have to type in that information.

The appropriate files will open, the index will load into memory, and you can begin your search. If an error occurs in opening the files, the program informs you of the type of error so you can correct it.

After Search opens the MAP index file and its associated source file, the name of the source file appears at the top of the screen. The blinking cursor on the second line under the letter E is a prompt for you to enter the key or keys you want to locate. Type the key(s) you're searching for and tap the enter key.

Each key must be three or more char-

Listing 3 continued

```
;
        LD      B,PENULN
        LD      C,0
        CALL    POSCUR          ;reposition cursor to next to last line
        POP     HL
        POP     DE
        POP     BC
        RET
;-------------------------------------------------------------
                                ;PRTREC
                                ;------------------------------
                                ;print record from screen
PRTREC  LD      C,0
        LD      A,(SORLN)
        CP      4
        JP      Z,PRTR2
        LD      A,(SORLN)
        DEC     A
PRTR2   LD      B,A
PRTRLP  LD      HL,PRTBUF
        LD      D,80
        PUSH    BC
        SVC     11              ;move screen line to buffer
        LD      HL,PRTBUF
        LD      B,80
        LD      C,0DH
        SVC     19              ;PRLINE
        POP     BC
        LD      A,22
        CP      B               ;has last line been reached?
        JP      Z,PRTR5
        JP      M,PRTR5
        INC     B
        JP      PRTRLP
PRTR5   RET
;-------------------------------------------------------------
                                ;JP2DOS
                                ;------------------------------
                                ;jp to TRSDOS without cmd
JP2DOS  SVC     36
;-------------------------------------------------------------
                                ;DOSCMD
                                ;------------------------------
                                ;jp to TRSDOS with cmd
                                ;execute CMD file then DOS
                                ;HL->name of CMD file
DOSCMD  PUSH    HL
        LD      B,0
DOSLP   LD      A,(HL)
        CP      0DH
        JP      Z,DOSEXC
        INC     HL
        INC     B
        JP      DOSLP
DOSEXC  POP     HL
        SVC     37              ;exc prog, then DOS
DOSLLP  NOP
        JP      DOSLLP          ;just in case
;-------------------------------------------------------------
                                ;DISBRK
                                ;------------------------------
DISBRK  LD      HL,0            ;disable break
        SVC     3
        LD      HL,DSBRK
        SVC     3               ;whenever break is pressed
                                ;program will jp to DSBRK
        RET
;-------------------------------------------------------------
                                ;DSBRK
                                ;------------------------------
DSBRK   PUSH    AF              ;break routine
        LD      A,'Y'
        LD      (BREAK),A
        POP     AF
        RET
BREAK   DEFB    'N'             ;break = Y, no break = N
;-------------------------------------------------------------
                                ;DIVIDE
                                ;------------------------------
DIVIDE  PUSH    BC              ;divide A into HL
        LD      B,1
        LD      C,A
        SVC     23
        LD      A,C
        POP     BC
        RET
;-------------------------------------------------------------
                                ;MULT
                                ;------------------------------
MULT    PUSH    BC              ;mult HL x A
        LD      B,0
        LD      C,A
        SVC     23
        LD      A,C
        POP     BC
        RET
;-------------------------------------------------------------
                                ;PTFS
                                ;------------------------------
                                ;point to file spec
PTFS    INC     HL              ;bypass char count
        CALL    EATNBNK         ;bypass cmd
        CALL    EATBNK          ;bypass trailing blanks
        RET                     ;HL -> first char of file spec
;-------------------------------------------------------------
                                ;ERRDSP
                                ;------------------------------
                                ;error dsp for Mod II TRSDOS
ERRDSP  NOP
        PUSH    BC
        LD      B,A
        LD      HL,MSGAREA
        SVC     52              ;ERRMSG
        LD      HL,MSGAREA
        CALL    VDLINE
        POP     BC
        RET
MSGAREA DEFS    80              ;error message area
        DEFB    03
;-------------------------------------------------------------
                                ;STMEND
                                ;------------------------------
STMEND  LD      (MEMEND),DE     ;store end of memory
        RET                     ;must be called before DE changed!!!!
MEMEND  DEFW    0               ;stores top address of mem available
;-------------------------------------------------------------
INDCB   DEFB    0               ;input (source) file DCB
IDCB1   DEFB    0
        DEFB    0
        DEFB    0
IDCBRL  DEFB    0               ;record length
        DEFB    0
        DEFB    0
IDCBCR  DEFB    0               ;current record
        DEFB    0
        DEFB    0
IDCBLR  DEFB    0               ;last record
        DEFB    0
IDCBOS  DEFB    0
        DEFS    50
INLLR   DEFW    0               ;true last sector
;
MAPDCB  DEFB    0               ;MAP file DCB
MDCB1   DEFB    0
        DEFB    0
        DEFB    0
MDCBRL  DEFB    0               ;record length
        DEFB    0
        DEFB    0
```

Listing 3 continued

✓ See List of Advertisers on Page 291

acters long, and no key or combination of keys can be longer than 60 characters. Search ignores capitalized letters and lets you search for partial keys. For example, the key "cdonal" locates records containing the names McDonald and MacDonald.

You can search for combinations of keys within records or sentences. You find combinations by using the logical operators AND or OR. AND is represented by the + symbol (hold the shift key and press +). OR is represented by the ¦ symbol (hold the control key and press zero on the Model II; hold the shift key and press @ on the Model III).

On the Model II the operators + and ¦ appear in inverse video to distinguish them from characters in a key. On the Model III a small, solid white rectangle appears under the operator to indicate that it isn't a character in a key. For example, the search key Jack + Jill¦Jack + nimble finds records that contain the names Jack and Jill or records containing the name Jack and the word nimble.

"You can search for combinations of keys within records or sentences."

On the Model II, enter the + character as part of a key by pressing F1 and enter the ¦ character as part of a key by pressing F2. On the Model III the @ key enters special characters from the keyboard.

If the key you're searching for has the symbol + or ¦, you can enter it in the following manner: press the @ key (unshifted) and the cursor becomes larger. Then enter either + or ¦. You enter the @ character into a key in the same manner.

After you enter a valid key, the program starts its search and the message "Searching" appears in the upper left corner of the screen. The number of each sector searched appears next to the word Searching.

When the program locates the first record in the file with the appropriate key (or keys), it displays the record at the bottom of the screen. One record appears at a time and the program waits for your instructions to continue the search, abort the search, or display text around the located record.

To continue the search, press the down-arrow key after each line of text appears. Pressing and holding the down-arrow key (and the repeat key on the Model II) lets you continuously scroll through records as the program finds them. When the program has

Listing 3 continued

```
MDCBCR  DEFB    0                    ;current record
        DEFB    0
        DEFB    0
MDCBLR  DEFB    0                    ;last record
        DEFB    0
MDCBOS  DEFB    0
        DEFS    50
MAPLLR  DEFW    0                    ;true last sector
;
OUTDCB  DEFS    60                   ;output file DCB
;
SORLN   DEFB    0                    ;start of record line number
                                     ;for each line of rec displayed
                                     ;(SORLN) is decremented
;-------------------------------------------------------------------
STMAP   DEFB    0
        END
```

```
        Source Line

;MINDEX3/SRC...1.2
;Mar 05, 1983
;-------------------------------------------------------------------
;MINDEX INDEXING PROGRAM FOR MODEL III
;MASTER SECTION: links main program with Model III I/O routines
                ;and general purpose routines
;COPYRIGHT 1983, SOFTSHELL CORPORATION   ALL RIGHTS RESERVED
;-------------------------------------------------------------------
PROGII  PSECT                       ;begin relocatable program section
        EXTERN  BEGIN               ;BEGIN = start of M3
START   JP      BEGIN
        LINK    'M3/REL'            ;main program
                                    ;note: the program is called M3 because it
                                    ;was originally developed on Model III
                                    ;In its current form it will work with
                                    ;Model II or III given the proper I/O
                                    ;module
        LINK    'GR/REL'            ;general programs
        LINK    'MODIII/REL'        ;Model III I/O routines
        END     START
```

Program Listing 4. Assembly-language listing of Mindex3/SRC.

```
        Source Line

;SEARCH3/SRC...1.2
;Mar 05, 1983
;-------------------------------------------------------------------
;SEARCH PROGRAM FOR MODEL III
;MASTER SECTION: links S3, GR, S3OPEN, S3DSP, S3OPT and MODIII
;COPYRIGHT 1983, SOFTSHELL CORPORATION   ALL RIGHTS RESERVED
;-------------------------------------------------------------------
PROGII  PSECT                       ;begin relocatable program section
;-------------------------------------------------------------------
        EXTERN  BEGIN               ;BEGIN = start of S3
;-------------------------------------------------------------------
START   JP      BEGIN
        LINK    'S3/REL'            ;SEARCH program proper
                                    ;note: the program is called
                                    ;S3 because it was originally
                                    ;developed on the Model III
                                    ;In its current form it will
                                    ;work with Model II or III
                                    ;given the proper I/O module
        LINK    'GR/REL'            ;general routines
        LINK    'S3OPEN/REL'        ;SEARCH open files routine
        LINK    'S3DSP/REL'         ;SEARCH display
        LINK    'S3OPT/REL'         ;SEARCH options
        LINK    'MODIII/REL'        ;Model III I/O routines
        END     START
```

Program Listing 5. Assembly-language listing of Search3/SRC:1.

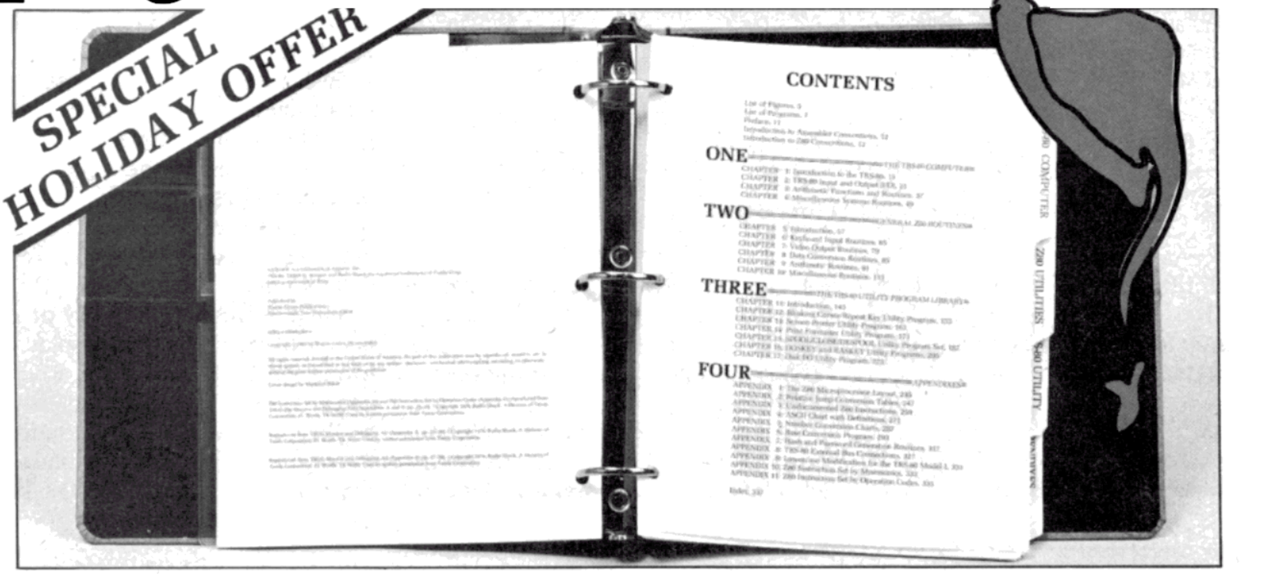
Program Listing 6. Assembly-language listing of MODIII/SRC.

```
        Source Line

;MODIII/SRC  VERSION 1.2
;Jun 26, 1983
;-------------------------------------------------------------------
;MAP MODEL III I/O ROUTINES, BUFFERS AND PARAMETERS
;COPYRIGHT 1983, SOFTSHELL CORPORATION  ALL RIGHTS RESERVED
;-------------------------------------------------------------------
MODIII  PSECT                       ;begin relocatable program section
;-------------------------------------------------------------------
                                    ;public routines
                                    ;--------------------------------
        PUBLIC  BOTTOM              ;position cursor to last line of screen
        PUBLIC  CLOSE               ;close file (DE->DCB)
        PUBLIC  CURPOS              ;determine cursor pos (coor:BC)
        PUBLIC  DISBRK              ;disable break (for Model II)
        PUBLIC  DIVIDE              ;divide A into HL
        PUBLIC  DOSCMD              ;jp to TRSDOS and execute command
        PUBLIC  ERASEL              ;erase to end of line
        PUBLIC  ERASES              ;erase to end of screen
        PUBLIC  ERRDSP              ;MAP error display routine
        PUBLIC  HOME                ;position cursor to top line of screen
        PUBLIC  INIT                ;initialize file (create/open, pos to 1st rec)
                                    ;                (DE->DCB, HL->rec buf)
        PUBLIC  JP2DOS              ;jp to TRSDOS
        PUBLIC  KBCHAR              ;get char (or null) from keyboard
        PUBLIC  KBINIT              ;initialize keyboard (Mod II)
        PUBLIC  KBLIN3              ;get line from keyboard and end with 03
        PUBLIC  KBLINE              ;get line from keyboard and end with 0DH
        PUBLIC  KBWAIT              ;wait for char from keyboard
        PUBLIC  KEYIN               ;get search key(s) form keyboard
        PUBLIC  MOVELN              ;move line from screen to buffer
                                    ;HL->buf, D=no. char, BC=cursor coor
        PUBLIC  MULT                ;multiply routine
        PUBLIC  NEWLN               ;role screen and pos cur to next to last line
        PUBLIC  OPEN                ;open file (DE->DCB, HL->record buffer)
        PUBLIC  OPINIT              ;open and initialize file
                                    ;          (DE->DCB, HL->record buffer)
        PUBLIC  POSCUR              ;position cursor (coordinates in BC)
        PUBLIC  POSEOF              ;position to end of file
        PUBLIC  POSN                ;position to record in file
        PUBLIC  PRINIT              ;initialize printer
        PUBLIC  PRLINE              ;print line (HL->buf, B=no. char, C=end char)
        PUBLIC  PRSCN               ;print entire screen
        PUBLIC  PRTBLN              ;print last line of text
        PUBLIC  PRTREC              ;print current record
        PUBLIC  PTFS                ;point to file spec in cmd line
        PUBLIC  READDR              ;read file: direct access
        PUBLIC  READNX              ;read next record in file
        PUBLIC  SCRPRO              ;set scroll protect line (A=line no.)
        PUBLIC  SETILR              ;set INLRR to true last record
        PUBLIC  SETMLR              ;set MAPLLR to true last record
        PUBLIC  STMEND              ;store top of memory in MEMEND
        PUBLIC  VDCHAR              ;display char in A register
        PUBLIC  VDCLS               ;clear screen
        PUBLIC  VDINIT              ;initialize video I/O (for Model II)
        PUBLIC  VDLIN3              ;display line (HL->msg; don't print 0DH)
        PUBLIC  VDLINE              ;display line (HL->msg)
        PUBLIC  WPKBWT              ;word processor kbwait routine
        PUBLIC  WRITE               ;direct write routine
        PUBLIC  WRITNX              ;write next record to file
;-------------------------------------------------------------------
                                    ;MODIII buffers
                                    ;--------------------------------
        PUBLIC  ACCESS              ;dummy for MODII compatability
        PUBLIC  INDCB               ;INput (source) file DCB
        PUBLIC  IDCBRL              ;Input file DCB Record Length
        PUBLIC  IDCBCR              ;Input file DCB Current Record
        PUBLIC  IDCBLR              ;Input file DCB Last Record
        PUBLIC  INLLR               ;true last record of source file
        PUBLIC  IDCBOS              ;Input file DCB xxxxxxx
        PUBLIC  MAPDCB              ;MAP (index) file DCB
        PUBLIC  MDCBRL              ;Map file DCB Record Length
        PUBLIC  MDCBCR              ;Map file DCB Current Record
        PUBLIC  MDCBLR              ;Map file DCB Last Record
        PUBLIC  MAPLLR              ;true last record of index file
        PUBLIC  MDCBOS              ;Map file DCB xxxxxxxx
        PUBLIC  MEMEND              ;contents ->top of memory
        PUBLIC  OUTCHR              ;record buffer (1 byte) for output file
        PUBLIC  OUTDCB              ;output file DCB
        PUBLIC  PRTBUF              ;printer buffer (80 char)
        PUBLIC  SORLN               ;start of record line no.
        PUBLIC  STMAP               ;start of MAP index file (always last byte
                                    ;of SEARCH program)
;-------------------------------------------------------------------
                                    ;MODIII I/O parm
                                    ;--------------------------------
        PUBLIC  ARROWD              ;arrow down
        PUBLIC  ARROWL              ;arrow left
        PUBLIC  ARROWR              ;arrow right
        PUBLIC  ARROWU              ;arrow up
```

Listing 6 continued

Every Z80 assembly-language programmer needs this book.



Programming in assembly language requires good tools. *TRS-80/Z80 Assembly Language Library*, a complete reference book on TRS-80 Model I assembly language, is the best tool you can find. In over 300-pages, 45 figures, and 75 program listings, author Craig A. Lindley explains the details of Model I hardware and software and shows you how to write programs that squeeze every bit of performance out of your computer. This book will teach you:

- how to use ROM and DOS routines in your own programs
- how to perform disk input/output
- how to access the video display and keyboard
- how to write easy-to-use programs
- how to perform arithmetic operations
- how to use the undocumented Z80 instructions

You'll also learn about disk operating systems, device handling, base conversion, parameter passing, and more.

TRS-80/Z80 Assembly Language Library contains a library of ready-to-run utility programs that are worth many times the book's cost. Included with the book are two Model I TRSDOS-compatible disks* containing utilities for printer formatting and spooling, single-key entry of strings, disk editing, base conversion, password encoding and decoding, and more. You get more than 15 valuable utilities, and the source code files are included, so you can examine, modify, and learn from every program.

Many of the programs also run on the Model III, and those that do not require only minor modification. One of the book's four sections is dedicated to routines and programs that run on any Z80-based system. No matter what Z80 computer you program, this book has something for you.

It's a book that's designed to be used. The text is set in large type, and the book is bound in an 8½- × 11-inch easel-backed binder that stands up next to your computer. So there's no squinting at tiny type and fighting to keep the book open.

The whole package—book, disks, and binder—is yours by mail for the **special holiday price of only $29.97.** And if you charge it, you can even call toll-free to order. Reserve your copy now, because after December 31, the price will be $34.97.

TRS-80/Z80 Assembly Language Library. It's the assembly-language book for the '80s.

*Disks do not contain a disk operating system: two disk drives or a disk copy utility are required to transfer the files.

Call **Toll-Free 1-800-258-5473** for credit card orders or send $29.97 plus $4.50 shipping and handling to Wayne Green Books, Retail Sales, Peterborough NH 03458. Dealer inquiries invited.

HOLIDAY SPECIAL

I need *TRS-80/Z80 Assembly Language Library.* 33DB8L

Send me _____ copies of *TRS-80/Z80 Assembly Language Library* (BK7395-01) @ $29.97 each. I have enclosed $4.50 per system for shipping and handling.

☐ MasterCard ☐ VISA ☐ AmEx ☐ payment enclosed

Card #______________________ Expires________

InterBank#________ Signature______________________

Name ______________________________

Address ______________________________

City______________________ State_____ Zip________

Wayne Green Books, Retail Sales, Peterborough, NH 03458

TRS-80/Z80 Assembly Language Library is a Wayne Green publication
TRS-80, TRSDOS, Model I, Model III are registered trademarks of the Radio Shack Division of Tandy Corp.
Z80 is a registered trademark of Zilog

Due to printer error, unavailable until November 1, 1983. We apologize for any inconvenience.

```
(C)hange File, (M)AP, (O)ptions or (T)RSDOS

xxxxxxxxx Format
Inset from Left Margin:            x
Search Display Mode: (C/R/L)  x
Print Mode: (C/M)                      x
Output File Name:           xxxxxxx/OUT
Output File Status: (C/R/A)    x
Output Mode: (C/M)                x
```

Figure 9. Search options screen.

searched the entire file, the message "Enter Key" appears again. You can now search for another key or combination of keys.

Search Options and Commands

While the "Enter Key" message is present and the cursor is in column 1, you can jump to the Search minimenu by pressing the break key. The following message appears on the top line: "(C)hange File, (M)AP, (O)ptions or (T)RSDOS".

Typing O gives you the Options screen shown in Fig. 9. You can bypass any question in the options menu by pressing the enter key. The program displays the meaning of abbreviations used in the current prompt at the bottom of the screen.

The first line informs you of the format used to index the file and you can't change it while searching a file.

The inset number (X) ranges from zero to 9. Insetting is similar to indenting except that the first line of a record is flush with the left margin, and subsequent lines are inset from the left margin. The number displayed across from the word inset is the value currently used to inset records. You can change it by typing any number from zero to nine and then pressing the enter key. The default inset is two.

While searching for a key or combination of keys, you can display the information continuously, or a record or line at a time. Type C and press the enter key to have records continuously displayed as the program finds them. Type R and press the enter key for

Listing 6 continued

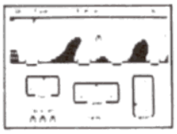
```
        PUBLIC  BRKEY           ;break key
        PUBLIC  BSPKEY          ;back space key
        PUBLIC  CPYPOS          ;copyright msg position
        PUBLIC  DELKEY          ;delete key
        PUBLIC  DFTFS           ;default file size
        PUBLIC  INSKEY          ;insert key
        PUBLIC  LSBYTE          ;col of last byte in 255 byte rec
        PUBLIC  PENULN          ;penultimate line (next to bottom)
        PUBLIC  SWIDTH          ;screen width
        PUBLIC  SWM1            ;screen width minus 1
        PUBLIC  TOPLN           ;top line of editor screen
;-------------------------------------------------------------
                                ;external routines and buffers
                                ;referenced by MODII
                                ;-----------------------------
        EXTERN  ANSWER          ;general purpose 1 byte buffer
        EXTERN  APPEND          ;append one string to another
        EXTERN  EATBNK          ;bypass leading blanks
        EXTERN  EATNBNK         ;bypass leading non-blank char
;-------------------------------------------------------------
                                ;Model III equates
                                ;-----------------------------
ARROWD  EQU     0AH             ;arrow down
ARROWL  EQU     08H             ;arrow left
ARROWR  EQU     09H             ;arrow right
ARROWU  EQU     5BH             ;arrow up
BRKKEY  EQU     1               ;BREAK
SWIDTH  EQU     64              ;screen width
SWM1    EQU     63              ;screen width minus 1
TOPLN   EQU     11              ;top line of edit display
LSBYTE  EQU     62              ;last col of record byte 255
PENULN  EQU     14              ;penultimate line (next to last line)
DFTFS   EQU     400             ;default 3by5 data file size
DELKEY  EQU     4               ;delete key (shift, down arrow, D)
INSKEY  EQU     1FH             ;insert key (CLEAR)
BSPKEY  EQU     2               ;backsp key (shift, down arrow, B)
CPYPOS  EQU     36              ;tab pos for copyright notice
;
KBWAIT  EQU     0049H
KBCHAR  EQU     002BH
VDCHAR  EQU     0033H
VDCLS   EQU     01C9H
INITIO  EQU     0069H
PRCHAR  EQU     003BH           ;print char
PRTSCR  EQU     01D9H           ;print screen
DINIT   EQU     4420H           ;initialize file
DOPEN   EQU     4424H           ;open file
DPOSN   EQU     4442H           ;position to rec in file
DREAD   EQU     4436H           ;direct read
DWRITE  EQU     4439H           ;direct write
DBKSPC  EQU     4445H           ;TRSDOS backspace routine
DPEOF   EQU     4448H           ;position to EOF
DCLOSE  EQU     4428H           ;close file
JP2DOS  EQU     402DH
CONDOS  EQU     429CH
CMDTXT  EQU     4225H
;
DELAY   EQU     0060H
CURSOR  EQU     4020H           ;cursor is at this position
CURCHR  EQU     4023H           ;cursor character
CURBLK  EQU     401CH           ;0 = blink, non0 = no blink
VIDEO   EQU     3C00H           ;start of video memory map
DSCRPR  EQU     4214H           ;scroll protect
DERDSP  EQU     4409H           ;TRSDOS error display routine
DIVIDE  EQU     4451H           ;TRSDOS divide routine
DMULT   EQU     444EH           ;TRSDOS multiple routine
EOMEM   EQU     4411H           ;end of memory
;-------------------------------------------------------------
                                ;HOME
                                ;-----------------------------
                                ;pos cursor to rl:cl
                                ;reg altered: HL
HOME    LD      HL,VIDEO
        LD      (CURSOR),HL
        RET
;-------------------------------------------------------------
                                ;BOTTOM
                                ;-----------------------------
                                ;pos cursor to bottom line, col 1
                                ;reg altered: HL
BOTTOM  LD      HL,VIDEO+3C0H
        LD      (CURSOR),HL
        RET
;-------------------------------------------------------------
                                ;NEWLN
                                ;-----------------------------
NEWLN   PUSH    BC              ;rolls screen and positions
        PUSH    DE              ;cursor to next to last line
        PUSH    HL
        CALL    BOTTOM          ;postion to bottom line
        LD      A,0DH
        CALL    VDCHAR          ;roll screen
        LD      BC,256*14
        CALL    POSCUR          ;position to next to last line
        LD      A,(SORLN)
        CP      3               ;has first line reached top of screen?
        JP      Z,NEWLN5        ;if it has do not dec (SORLN)
        LD      A,(SORLN)
        DEC     A               ;dec (SORLN)
        LD      (SORLN),A
NEWLN5  POP     HL
        POP     DE
        POP     BC
        RET
;-------------------------------------------------------------
                                ;CURPOS
                                ;-----------------------------
                                ;get cursor position
                                ;exit B=row-1, C=col-1
                                ;reg altered: AF,BC
CURPOS  PUSH    DE
        PUSH    HL
        LD      BC,0
        LD      HL,(CURSOR)
        LD      DE,VIDEO
        OR      A
        SBC     HL,DE
        JP      Z,CUR8
CURLP   PUSH    HL
        LD      DE,64
        OR      A
        SBC     HL,DE
        JP      Z,CUR5
        JP      C,CUR6
        INC     B
        POP     DE
        JP      CURLP
CUR5    POP     DE
        INC     B
        JP      CUR8
CUR6    POP     HL
        LD      C,L
CUR8    POP     HL
        POP     DE
        PUSH    BC
        POP     HL
        RET
;-------------------------------------------------------------
                                ;POSCUR
                                ;-----------------------------
                                ;position cursor
                                ;entry: B=row-1, C=col-1
POSCUR  PUSH    DE
        PUSH    HL
        LD      A,15            ;bottom line
        CP      B
        JP      C,POS8          ;exit if B > bottom line
        LD      A,63            ;last col on screen
        CP      C
        JP      C,POS8          ;exit if C > last col
        LD      HL,VIDEO        ;HL->start of video map
POSLP   LD      A,B             ;HL=HL+64 (repeat as needed)
        CP      0
        JP      Z,POS5
        LD      DE,64
        ADD     HL,DE
        DEC     B
        JP      POSLP           ;loop until B=0
POS5    LD      E,C             ;now set col value
        LD      D,0
        ADD     HL,DE
        LD      (CURSOR),HL     ;store new value in CURSOR
POS8    POP     HL
        POP     DE
        RET
;-------------------------------------------------------------
                                ;KBINIT
                                ;-----------------------------
KBINIT  RET                     ;no MODIII equivalent
;-------------------------------------------------------------
                                ;OPEN
                                ;-----------------------------
                                ;open file
                                ;entry: HL-> buffer
                                ;       DE->DCB
```

Listing 6 continued

THE PROGRAM STORE

NOW SAVE 20% TO 50% ON THESE HOLIDAY SPECIALS
SALE ENDS JAN. 10, 1984

LIMITED TO STOCK ON HAND FOR MODEL I AND III

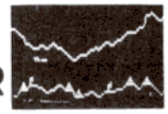


ALIEN TAXI

by John Olsen, Jr.
from Fantastic Software

You pilot the Alien Taxi against gravity to pick up and deliver passengers to an underground resort hotel. On the first level, you pick up a fare at each of the 12 taxi stands. Complete that and 12 more are on the second level. This skillful game for 1 or 2 players has sound effects. Joystick compatible.

#27188 16K Tape ~~$15.95~~ $12.76
#36515 32K Disk ~~$19.95~~ $15.96
20% OFF



PENETRATOR

from Melbourne House

Armed with missiles and bombs, you must fly your fighter to the enemy's cache of neutron bombs and destroy them. Your mission is in four stages, involving rugged terrain, caverns and manmade obstacles — not to mention enemy radar missiles and paratroopers. This new departure in arcade gaming allows you to set up your own terrain and enemy emplacements then save them for future use. Make your mission as hard or easy as you like. Joystick compatible.

#21160 16K Tape ~~$24.95~~ $19.96
#21182 32K Disk ~~$24.95~~ $19.96
20% OFF

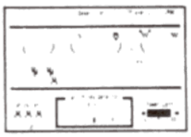
INVADERS FROM SPACE

by Carl Miller
from Acorn

A fast machine language approach to this classic (and addictive) space game. The aliens drop bombs and move from side to side trying to overrun your bases. You choose the speed, enemy bomb frequency and accuracy, your number of shots on screen and bases. Unlike most such games, you can move your base and simultaneously fire at the invaders. Full sound effects add even more excitement to the incredible action. Fun for all ages and skill levels.

#16591 16K Tape ~~$14.95~~ $11.96
#16603 32K Disk ~~$20.95~~ $16.76
20% OFF

DESERT PERIL



by John R. Olsen, Jr.
from Fantastic Software

Your people's only chance to survive is for you to cross the Great Desert or they will die from the Zagon's deadly virus. Arcade game with sound and graphics for 1 or 2 players has top ten scores and pause features. Use keys or joystick to dodge killer satellites, drone bomber balloons and flying dragons. Your skill and daring determine the future!

#29270 16K Tape ~~$15.95~~ $12.76
#29337 32K Disk ~~$19.95~~ $15.96
20% OFF

STRIKE FORCE

from Melbourne House

You are the warrior-savior of cities under relentless alien attack. You're armed with rapid-fire missiles, radar and incendiary star-shells, but you'll need your fastest thinking and instantaneous response to protect yourself and your people. Great real-time graphics!

#38809 16K Tape ~~$15.95~~ $11.16
#38618 32K Disk ~~$19.95~~ $13.96
30% OFF

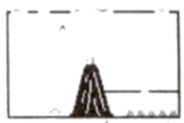
KING OF THE JUNGLE

by Vandewalle
from Acorn

An unusual combination of fast pinball action plus the challenges and dangers of an arcade game. Keep the ball in play, scoring points as it caroms from obstacle to obstacle, while wandering through the jungle searching for the hidden crown. And try to avoid the deadly Cobra. Lots of action, sound and fun! Choice of 3 screens. Can be used with Trisstick joystick.

#39080 16K Tape ~~$19.95~~ $9.96
#39114 16K Disk ~~$19.95~~ $9.96
50% OFF

DEVIL'S TOWER



by John Olsen Jr.
from Fantastic Software

It's up to you to save your planet from the aliens. They've reached the other side of DEVIL'S TOWER. Robot Scouts come at you from the mountain, their war machines fire from the valley and their protector ships put up force fields to protect them. Your skill and reflexes may protect you. Machine language, good graphics and sound effects. For 1 or 2 players.

#43366 16K Tape ~~$15.95~~ $12.76
#43377 32K Disk ~~$19.95~~ $15.96
20% OFF

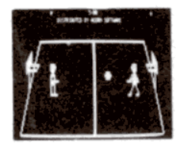
MARTIAN PATROL

by Rick Maurice
from Melbourne House

Prepare for the mission only the toughest survive! Control your patrol craft across security perimeter into hostile martian terrain riddled with craters and outcrops to jump over. Negotiate with extreme caution to avoid a devastating collision. Take the offensive against savage attacks by mutating alien crafts. The dreaded MARTIAN PATROL — the most feared mission on the planet Mars can be yours!

#34759 16K Tape ~~$15.95~~ $12.76
#34760 32K Disk ~~$19.95~~ $15.96
20% OFF

BASKET BALL

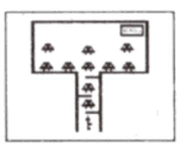


by John Allen
from Acorn

You have to be fast to keep up with the action as you try to outscore your opponent in five minutes of one-on-one basketball. Compete against a friend or your computer.

Steal the ball, duck around your opponent and slant toward the basket for a lay up! The graphics are based on a 3-dimensional depiction of a basketball court, and ball dribbling sounds add to the realism.

#16614 16K Tape ~~$14.95~~ $7.46
#16625 32K Disk ~~$20.95~~ $10.46
50% OFF



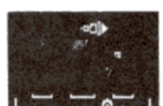
VENTURE

by Philip Case
from Horizons

Set off on a journey into the stronghold of an ancient wizard. Danger is everywhere, and the penalty for a mistake may be complete destruction! The ultimate challenge lies in the forbidden Riddle Room — are you ready for the VENTURE? With sound effects; joystick compatible.

#36111 16K Tape ~~$14.95~~ $10.46
#36122 16K Disk ~~$19.95~~ $13.96
30% OFF

WEERD

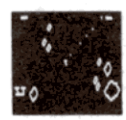


by Arthur Gleckler
from Big Five

Maybe you think you've seen some pretty strange things on your computer screen already, but you haven't seen ANYTHING until you've tried WEERD. It's you against the WEERDEST set of space craft ever. But don't just stare at them, get your shields up before they wipe out your planet. Disk version has voice sound effects and permanent high scores. Both are trisstick compatible.

#43388 16K Tape or ~~$19.95~~ $13.96
#43399 32K Disk ~~$19.95~~ $13.96
30% OFF

SUPER NOVA



by Hogue & Konyu
from Big Five

Asteroids surround your ship. You must shoot the asteroids, as well as any of the five types of alien spaceships. Use your thrusters for full movement and rotation of your ship — If you are overwhelmed, you can even jump to hyperspace! Written in fast machine code with superb graphics, this game is GREAT! Joystick compatible.

#18043 16K Tape ~~$15.95~~ $11.16
#18111 32K Disk ~~$19.95~~ $13.96
30% OFF

Over 2500 Programs for TRS-80, ATARI 400/800, APPLE, IBM, VIC 20 & C64

VISIT OUR STORES

Southwest Plaza, 8501 West Bowles Ave., Littleton, CO
Tenley Mall, 4200 Wisconsin Ave., NW, Washington, DC
Harvard Square, 13 Dunster St. Cambridge, MA
W. Bell Plaza, 6600 Security Blvd, Baltimore, MD
White Flint Mall, Rockville Pike, Rockville, MD
Cherry Hill Mall, Rt 38 & Haddonfield Rd, Cherry Hill, NJ
Monmouth Mall, Rt 35 & Wyckoff Rd, Eatontown, NJ
Willowbrook Mall, 1400 Willowbrook, Wayne, NJ
Nanuet Mall, Rt 59 & Middletown Rd, Nanuet, NY
Olentangy Plaza, 829 Bethel Rd, Columbus, OH
Westmoreland Mall, Rt. 30 East, Greensburg, PA
Montgomery Mall, 230 Montgomery Mall, North Wales, PA
Plymouth Meeting Mall, 500 Germantown Pike, Plymouth Meeting, PA
Century III Mall, SR 51 & SR 885, West Mifflin Borough, PA
Seven Corners Center, Falls Church, VA

COMING SOON TO:
CHICAGO
OKLAHOMA CITY
ST. PETERSBURG
TAMPA
TULSA

FRANCHISE OPENINGS IN SELECTED CITIES

To Order Call Toll Free 800-424-2738 For Information Call (703) 556-9778

Mailorders: List computer, item, item #, qty, tape/disk, rom, book, price, shipping, tax & total
Send check or M.O. for total purchase plus $2.00 shipping and handling. VA add 4% sales tax
Charge cards — include all embossed information. List name, address, city, state, zip & phone

SEND 25¢ FOR OUR COMPLETE COLOR COMPUTER CATALOG

✓ 80

THE PROGRAM STORE®

THE PROGRAM STORE Dept. 14-12-3 Box 9582 4200 Wisconsin Avenue, NW Washington, D.C. 20016

Listing 6 continued

```
                                    ;exit:  A=TRSDOS error code
OPEN    CALL    DOPEN
        PUSH    AF
        LD      A,0
        LD      (IDCBRL),A          ;RL=256
        LD      A,32
        LD      (IDCB1),A           ;RL=256??
        POP     AF
        RET
;------------------------------------------------------------
                                    ;INIT
                                    ;------------------------------
                                    ;initialize file
                                    ;DE->DCB
INIT    CALL    DINIT
        RET
;------------------------------------------------------------
                                    ;OPINIT
                                    ;------------------------------
                                    ;open file if present, else init file
                                    ;entry: HL->buffer
                                    ;       DE->DCB
                                    ;exit:  A=TRSDOS error code
OPINIT  PUSH    BC
        PUSH    DE
        PUSH    HL
        CALL    DOPEN               ;try to open file first
                                    ;keeps present record no.
        JP      Z,OPIT4
        POP     HL
        POP     DE
        POP     BC
        CALL    DINIT               ;sets last rec no. = 0
        RET
OPIT4   POP     HL                  ;open attempt successful
        POP     DE
        POP     BC
        RET
OUTCHR  DEFB    0                   ;output record buffer for Mod II
;------------------------------------------------------------
                                    ;CLOSE
                                    ;------------------------------
                                    ;close file
                                    ;entry: DE ->DBC
CLOSE   CALL    DCLOSE
        RET
;------------------------------------------------------------
                                    ;SETMLR
                                    ;------------------------------
                                    ;set MAPLLR to true last rec
SETMLR  LD      HL,(MDCBLR)
        DEC     HL                  ;MODIII adjustment
        LD      (MAPLLR),HL
        RET
;------------------------------------------------------------
                                    ;SETILR
                                    ;------------------------------
                                    ;set INLLR to true last record
SETILR  LD      HL,(IDCBLR)
        LD      A,(IDCBOS)
        CP      0
        JP      NZ,SETI4
        DEC     HL                  ;file ends at end of sector
                                    ;therefore on Mod III, must dec (IDCBLR)
                                    ;for true last record no.
SETI4   LD      (INLLR),HL
        RET
;------------------------------------------------------------
                                    ;POSN
                                    ;------------------------------
                                    ;position to rec in file
                                    ;entry: DE->DCB
                                    ;       BC=rec no.
                                    ;exit:  A=TRSDOS error code
POSN    CALL    DPOSN
        RET
;------------------------------------------------------------
                                    ;POSEOF
                                    ;------------------------------
                                    ;position to end of file
                                    ;entry: DE->DCB
                                    ;exit:  TRSDOS error code
POSEOF  CALL    DPEOF
        CALL    DREAD
        RET
;------------------------------------------------------------
                                    ;READDR
                                    ;------------------------------
                                    ;direct read
                                    ;entry: DE->DCB
                                    ;       BC=rec. no.
                                    ;exit:  A=TRSDOS error msg
READDR  PUSH    BC
        CALL    DPOSN               ;position to rec BC
        JP      NZ,RDDR5            ;jp if error
        CALL    DREAD               ;direct read
        POP     BC
        PUSH    AF                  ;temp store any error msg code
        CALL    DPOSN               ;reposition to rec=BC
                                    ;needed for Model II compatibility
        POP     AF
        RET
RDDR5   POP     BC
        RET
;------------------------------------------------------------
                                    ;READNX
                                    ;------------------------------
                                    ;read next record
                                    ;entry: DE->DCB
                                    ;exit:  A=TRSDOS error code
READNX  CALL    DREAD
        RET
;------------------------------------------------------------
                                    ;WRITE
                                    ;------------------------------
                                    ;direct write
                                    ;entry: DE->DCB
                                    ;       BC=rec. no.
WRITE   PUSH    BC
        PUSH    DE
        CALL    DPOSN               ;position to record
        POP     DE
        PUSH    DE
        CALL    DWRITE              ;write
        POP     DE
        POP     BC
        CALL    DPOSN               ;reposition to BC
        RET
;------------------------------------------------------------
                                    ;WRITNX
                                    ;------------------------------
                                    ;sequential write
                                    ;entry: DE->DCB
                                    ;exit:  A=TRSDOS error code
WRITNX  CALL    DWRITE
        RET
;------------------------------------------------------------
                                    ;WPKBWT
                                    ;------------------------------
                                    ;word processing KBWAIT routine
                                    ;cursor alternates with character
```

Listing 6 continued

record scroll.

As it locates and displays each record containing the keys, the program waits for your instruction to continue, abort the search, print the record, or output the record to another disk file. Type L and press the enter key for line scroll to have Search pause after displaying each line. The default search display mode is record scroll.

You can print records found during a search continuously or manually. Type C and press the enter key to have each line printed as it appears on the screen. Type M and press the enter key to let you selectively print records or lines of text as they appear on the screen. The Search Display Mode determines whether a line or the entire record is printed. The default print mode is manual.

The output file name is the name of the file to which you want to transfer records. Enter the file name using standard TRSDOS notation to designate the name, extension, disk drive, and password. If you don't specify an output file name, the output file has the same specifications as the source file except for the extension OUT.

You can close, reset, or open the output file. Type C and press the enter key to close an output file. Type R and press the enter key to reset the output file to the first sector. If the file is closed, this opens the file and sets it to the first sector.

Type A and press the enter key to append additional records to a file. The program retains previously written records. If the program finds the file closed, it opens it and sets the file pointer to the end-of-file marker. Using append on a newly created file is equivalent to resetting the file. The default output file status is closed.

You can write records found during a search to the output file continuously or manually. Type C and press the enter key to have each record written to the file as it's found. Type M and press the enter key to manually select records to be written to the output file. The default output file mode is manual.

While searching you might wish to display the text around a particular record. Tap L to call the list routine. A dotted line indicates that you changed from the search to the list mode. The message on the top line reads "Listing XXXXX", where XXXXX is the sector number. In the list mode, the program sets the inset to zero while word wraparound remains in effect.

Press the down-arrow key to display a line at a time. If you hold the down-arrow key (and the repeat key on the Model II), the file scrolls forward.

To backspace through the source file, press the up-arrow key and the text jumps back three sectors. The message "*Backspace Three Sectors*" appears. You can backspace to the beginning of the file in this manner.

While listing, you can continue your search where you left off by pressing the S key. A dotted line indicates that you've changed from listing to searching the source file. The message on the top line of the screen reads "Searching XXXXX".

You can write records found during searching or listing to a separate output file. Three of the options deal with the output file.

The default output file specification is the same as the index file but has the extension OUT. If you don't want to change the name, tap the enter key in response to the Output File Name prompt.

Typing R and pressing the enter key opens the output file and sets the output file pointer to the first record. Typing A and pressing the enter key opens the output file and sets the file pointer to the end of the file so that you can append records to the file.

To close an open file, type C and press the enter key in response to the prompt. If the file is already closed, the program ignores the command. To bypass the output file status prompt, press the enter key and the status won't change.

You can write records to the output file as they appear by typing C and pressing the enter key in response to the output mode prompt. Alternatively, you can select which records you want to copy to the output file by typing M and pressing the enter key in response to the output mode prompt.

As each record appears, you can copy it to the output file by pressing O before going on to the next record. Records are written to the output file in the same format they have in the source file.

With MAP, you can edit or index and search the output file. At the end of the search, either close the file (using the options menu) or exit from the search program (the output file is closed whenever you exit Search).

You can print the entire contents of the screen by pressing W any time the program searches or lists lines of the source file on the screen. If the Search Display mode is set for record display, you can print the last record displayed any time while searching by pressing P.

If the Search Display mode is set for line display, you can print the display's bottom line at any time while searching by pressing the P key. If you do so and

Listing 6 continued

```
WPKBWT  PUSH    BC              ;BC used as counter
        LD      A,1
        LD      (CURBLK),A      ;non blinking cursor
        LD      A,143
        LD      (CURCHR),A      ;large cursor over character
WPKB3   LD      BC,400
        LD      A,0EH           ;cursor on
        CALL    VDCHAR
WPKB6   CALL    KBCHAR
        CP      0
        JP      NZ,WPKB20       ;jp if char entered
        DEC     BC
        LD      A,B
        OR      C
        CP      0
        JP      NZ,WPKB6
;...........................
        LD      A,0FH           ;cursor off
        CALL    VDCHAR
        LD      BC,500
WPKB12  CALL    KBCHAR
        CP      0
        JP      NZ,WPKB20       ;jp if char entered
        DEC     BC
        LD      A,B
        OR      C
        CP      0
        JP      NZ,WPKB12
        JP      WPKB3
;
WPKB20  PUSH    AF
        LD      A,0FH
        CALL    VDCHAR
        LD      A,0
        LD      (CURBLK),A      ;blinking cursor
        LD      A,176
        LD      (CURCHR),A
        POP     AF
        POP     BC
        RET
;---------------------------------------------------------------
                                ;KBLIN3 and KBLINE
                                ;-------------------------------
KBLIN3  NOP                     ;EOR=03
        LD      A,03
        JR      KBL03
KBLINE  NOP                     ;entry: HL->buf,B= max length
                                ;exit: B=length not including EOR char
                                ;EOR=0DH or 03
                                ;returns with last char entered in A
        LD      A,0DH
KBL03   LD      (EORCHR),A
        LD      A,0EH
        CALL    VDCHAR          ;cursor on
        LD      C,0
KBL05   CALL    KBWAIT
        LD      D,A
                                ;key entered
        CP      0DH             ;enter?
        JP      Z,KBL0DH
        CP      08              ;backspace?
        JP      Z,KBLBSP
        CP      BRKKEY          ;break?
        JP      Z,KBBRK
        CP      20H             ;' '?
        JP      M,KBL05         ;jp if control char
        LD      A,B
        CP      C               ;have max char's been entered?
        JP      Z,KBL05
        LD      A,D             ;reload since A used for calc
        LD      (HL),A          ;store in buf
        INC     C               ;inc char count
        INC     HL
        CALL    VDCHAR          ;display char
        JP      KBL05           ;loop until EOR
KBLBSP  NOP                     ;back space routine
        LD      A,0
        CP      C
        JP      Z,KBL05         ;do nothing if start of buf
        LD      A,8
        CALL    VDCHAR          ;backspace
        DEC     HL              ;dec buf pointer
        DEC     C               ;dec char count
        JP      KBL05
KBBRK   LD      A,(EORCHR)      ;break key entered
        LD      (HL),A
        CALL    VDCHAR
        LD      A,BRKKEY
        LD      (EORCHR),A
        JP      KBL0DI
KBL0DH  NOP                     ;place EOR (03 or 0DH)
        LD      A,(EORCHR)
        LD      (HL),A
;       CALL    VDCHAR          ;display EOR   VOID VOID VOID VOID VOID
KBL0DI  INC     HL              ;HL->byte past EOR
        LD      A,0FH
        CALL    VDCHAR          ;cursor off
        LD      A,(EORCHR)
        LD      B,C
        RET
EORCHR  DEFB    0DH             ;EOR character
;---------------------------------------------------------------
                                ;PRINIT
                                ;-------------------------------
PRINIT  RET                     ;initialize printer (MOD II only)
;---------------------------------------------------------------
                                ;SCRPRO
                                ;-------------------------------
SCRPRO  LD      (DSCRPR),A      ;scroll protect
        RET
;---------------------------------------------------------------
                                ;ERASES
                                ;-------------------------------
                                ;erase to end of screen
ERASES  LD      A,1FH
        CALL    VDCHAR
        RET
;---------------------------------------------------------------
                                ;ERASEL
                                ;-------------------------------
                                ;erase to end of line
ERASEL  LD      A,1EH
        CALL    VDCHAR
        RET
;---------------------------------------------------------------
                                ;VDINIT
                                ;-------------------------------
VDINIT  LD      A,0             ;clear screen and initialize
        CALL    SCRPRO
        CALL    ERASES
        RET
;---------------------------------------------------------------
                                ;VDLINE
                                ;-------------------------------
                                ;display line
VDLINE  NOP                     ;entry: HL->string
                                ;0DH or 03 stop display
                                ;exit: HL ->next char
VDL05   LD      A,(HL)          ;get char
        CP      0DH             ;EOR?
        JR      Z,VDL12
        CP      03              ;EOR?
        JR      Z,VDL12
        CALL    VDCHAR          ;display char
        INC     HL
        JR      VDL05           ;loop until EOR
VDL12   CALL    VDCHAR          ;display EOR char
        INC     HL              ;-> next char
        RET
;---------------------------------------------------------------
                                ;VDLIN3
                                ;-------------------------------
                                ;display line, but not 0DH
VDLIN3  NOP                     ;entry: HL->string
                                ;0DH or 03 stop display
                                ;DOES NOT PRINT 0DH!
                                ;exit: HL ->next char
VDL305  LD      A,(HL)
        CP      0DH
        JR      Z,VDL312
        CP      03
        JR      Z,VDL312
        CALL    VDCHAR
        INC     HL
        JR      VDL305
VDL312  INC     HL              ;-> next char
        RET
;---------------------------------------------------------------
                                ;KEYIN
                                ;-------------------------------
                                ;get search key (includes +/! op)
                                ;entry: HL -> input buffer
                                ;       B = max no. of char
KEYIN   LD      A,0EH
        CALL    VDCHAR          ;turn cursor on
        CALL    ERASEL          ;erase to end of line
        LD      C,0             ;char counter
GETCHR  NOP
        CALL    KBWAIT          ;get char from keyboard
        CP      20H             ;char or control?
        JP      M,CTRL          ;control if less 20H
        CP      '@'
        JP      Z,KBCMD         ;jp KBCMD if cmd (i.e. '@' pressed)
        JP      ALPHA           ;alphanumeric entered
;...............................................................
ENTAND  LD      A,C
        CP      B
        JP      Z,GETCHR
        INC     C
        LD      A,'+'
        JP      ALPHA5
ENTOR   LD      A,C
        CP      B
        JP      Z,GETCHR
        INC     C
        LD      A,124
        JP      ALPHA5
CMDAND  LD      A,253           ;and code
        LD      (HL),A          ;store and cmd
        LD      A,'+'           ;and char
        CALL    VDCHAR
        CALL    MARK            ;mark as operator
        INC     HL
        JP      GETCHR
CMDOR   LD      A,254           ;or code
        LD      (HL),A
        LD      A,'|'           ;or char
        CALL    VDCHAR
        CALL    MARK            ;mark as operator
        INC     HL
        JP      GETCHR          ;loop
;
                                ;alpha numeric char
ALPHA   LD      D,A             ;temp store A
        LD      A,C             ;A now = char count
        CP      B               ;cp to max char
        JP      Z,GETCHR        ;loop until CTR or CMD
                                ;char ok so process
        INC     C               ;bump counter
        LD      A,D             ;restore A
        CP      '+'
        JP      Z,CMDAND
        CP      60H             ;reverse '
        JP      Z,CMDOR
ALPHA5  LD      (HL),A
ALPHA7  INC     HL              ;bump HL
        CALL    VDCHAR          ;display char
        JP      GETCHR          ;loop
;...............................................................
```

Listing 6 continued

✓ See List of Advertisers on Page 291

the printer isn't ready, the program pauses until you press the break key.

You can exit from the Search program by pressing the break key while it lists or searches text, or by pressing the break key in response to the "Enter Key" message. The minimenu appears again on the top line. Press T to return to TRSDOS.

Possible Applications

Entering reference collections in disk files and searching with Search eliminates extensive cross-indexed card files or specially coded computer files. You can enter titles, authors, or journal names in any order. You can also add key words to the record.

Search's output option is helpful when composing the reference section of any research or technical paper. Search your reference files for the appropriate references and copy them to an output file. You can incorporate the output file into a manuscript or edit it as an independent file.

You can keep information about clients or club members in text format and retrieve it rapidly with Search. A realtor could keep a record of each client, type of property required, geographic location, and price range. When a new property becomes available, he could search for the appropriate keys and locate the clients who might be interested. He could also use the program to select properties a new client might want to purchase.

You can search parts listings rapidly. Each record of a file could store information on automobile parts, and contain a description of part, stock num-

Listing 6 continued

```
CTRL    NOP                     ;control char entered
        CP      0DH             ;? CR
        JP      Z,CTRL0D
        CP      08              ;? backspace
        JP      Z,CTRL08
        CP      1               ;was break hit?
        JP      Z,CTRL01
        JP      GETCHR          ;other CTRL not allowed
CTRL01  NOP                     ;break key routine
        LD      A,C
        CP      0               ;was break hit on ct = 0?
        JP      NZ,CTRL02       ;start over
        LD      A,0FH
        CALL    VDCHAR          ;turn off cursor
        LD      A,BRKKEY        ;A=1 => break hit
        LD      B,0
        RET
CTRL02  LD      A,BRKKEY
CTRL0D  NOP                     ;EOR
        LD      (HL),A
        JP      KBEND
CTRL08  NOP                     ;backspace routine
        LD      D,A             ;temp save A in D
        LD      A,C
        CP      0               ;if zero cannot backspace
        JP      Z,GETCHR
        DEC     C               ;dec counter
        DEC     HL              ;dec buffer pointer
        LD      A,08
        CALL    VDCHAR          ;erase one char
;
        PUSH    HL
        LD      HL,(CURSOR)
        LD      DE,64
        OR      A
        ADD     HL,DE
        LD      A,'-'
        LD      (HL),A
        POP     HL
        JP      GETCHR
;..............................................................
KBCMD   NOP                     ;keyboard command routine
        LD      A,143
        LD      (CURCHR),A      ;large cursor
        CALL    KBWAIT          ;get char
        CP      0DH
        JP      Z,RSTCUR
        LD      D,A
        LD      A,C
        CP      B
        JP      Z,RSTCUR
        LD      A,D             ;restore A
        CP      '+'
        JP      Z,KBAND
        CP      60H             ;reverse '
        JP      Z,KBOR
        CP      '@'
        JP      Z,CMDCHR
CMDCHR  NOP
        LD      (HL),A
        CALL    VDCHAR
        JP      CMDOK
KBAND   LD      A,'+'
        LD      (HL),A
        CALL    VDCHAR
        JP      CMDOK
KBOR    LD      A,'|'
        LD      (HL),A
        CALL    VDCHAR
CMDOK   NOP
        INC     C
        INC     HL
RSTCUR  LD      A,176
        LD      (CURCHR),A
        JP      GETCHR
;
KBEND   LD      B,C
        PUSH    AF
        LD      A,0FH
        CALL    VDCHAR          ;turn off cursor
        POP     AF              ;A = EOR character
        RET
;..............................................................
MARK    PUSH    HL              ;place mark on line under char
        LD      DE,63
        OR      A
        ADD     HL,DE
        LD      A,130
        LD      (HL),A
        POP     HL
        RET
;--------------------------------------------------------------
                                ;MOVELN
                                ;-------------------------------
                                ;move char from screen to buffer
                                ;entry: HL->buffer, D= no. char
                                ;       BC=cursor coordinates
MOVELN  LD      A,D
        CP      0
        RET     Z               ;no transfer if char count = 0
        LD      (COOR),BC       ;store cursor coordinates
        CALL    CURPOS          ;position cursor
        PUSH    BC
        LD      BC,(COOR)
        CALL    POSCUR
        LD      B,0
        LD      C,D
        PUSH    BC
        LD      D,H
        LD      E,L
        LD      HL,(CURSOR)
        POP     BC
        LD      DE,PRTBUF
        LDIR                    ;transfer to PRTBUF
        POP     BC
        CALL    POSCUR          ;reposition cursor
        RET
;
COOR    DEFW    0               ;coordinate storage
;--------------------------------------------------------------
                                ;PRLINE
                                ;-------------------------------
PRLINE  NOP                     ;print line
                                ;HL->buffer
                                ;B=no. char, C=ending control char
PRLLP   LD      A,(HL)
        CALL    PRCHAR          ;print char
        CALL    KBCHAR          ;see if break pressed
        CP      01
        RET     Z
        INC     HL              ;pt to next char
        LD      A,B
        CP      0
        JP      Z,PRLNED
        DEC     B               ;dec char count
        JP      PRLLP
PRLNED  LD      A,C
        CALL    PRCHAR
        RET
;--------------------------------------------------------------
                                ;PRSCN
                                ;-------------------------------
                                ;print screen
PRSCN   CALL    PRTSCR
        RET
;--------------------------------------------------------------
PRTBUF  DEFS    80              ;printer buffer
        DEFB    0DH
;--------------------------------------------------------------
                                ;PRTBLN
                                ;-------------------------------
PRTBLN  PUSH    BC              ;print next to bottom line
        PUSH    DE
        PUSH    HL
        LD      B,PENULN
        LD      C,0
        LD      D,SWIDTH
        LD      HL,PRTBUF
        CALL    MOVELN
        LD      HL,PRTBUF
        LD      B,SWIDTH
        LD      C,0DH
        CALL    PRLINE
        LD      B,PENULN
        LD      C,0
        CALL    POSCUR          ;reposition cursor to next to last line
        POP     HL
        POP     DE
        POP     BC
        RET
;--------------------------------------------------------------
                                ;PRTREC
                                ;-------------------------------
PRTREC  NOP                     ;print record from screen
        LD      C,0
        LD      A,(SORLN)
        CP      3
        JP      Z,PRTR2
        LD      A,(SORLN)
        DEC     A
PRTR2   LD      B,A
PRTRLP  LD      HL,PRTBUF
        LD      D,SWIDTH
        PUSH    BC
        CALL    MOVELN
        LD      HL,PRTBUF
        LD      B,SWIDTH
        LD      C,0DH
        CALL    PRLINE
        POP     BC
        LD      A,PENULN
        CP      B               ;has last line been reached?
        JP      Z,PRTR5
        JP      M,PRTR5
        INC     B
        JR      PRTRLP
PRTR5   RET
;--------------------------------------------------------------
                                ;MULT
                                ;-------------------------------
                                ;entry: HL=multiplicand
                                ;       A=multiplier
                                ;exit:  HL=product, A=overflow byte
MULT    CALL    DMULT
        PUSH    HL
        LD      H,L
        LD      L,A
        POP     AF              ;A now has over flow byte
        RET
;--------------------------------------------------------------
                                ;PTFS
                                ;-------------------------------
PTFS    LD      HL,CMDTXT       ;point (HL) to cmd line
        CALL    EATNBNK         ;bypass cmd
        CALL    EATBNK          ;bypass trailing blanks
        RET                     ;HL-> first char of file spec
;--------------------------------------------------------------
                                ;DOSCMD
                                ;-------------------------------
                                ;execute CMD file then DOS
                                ;HL->name of CMD file
DOSCMD  PUSH    HL
        LD      DE,CMDTXT       ;DE->TRSDOS cmd line buffer
        CALL    APPEND          ;move cmd to cmd line buffer
                                ;needed to pass parameters to program
        POP     HL
        JP      COMDOS
```

Listing 6 continued

See List of Advertisers on Page 291

ber, manufacturer, model number, and equivalent products from other manufacturers. A customer needs a water pump for a 1970, 250-cubic-inch Chevrolet. Type in Chevro + 1970 + 250ci and the appropriate records appear with the model number, your stock number, and price.

Limitations

The Mindex and Search programs have several limitations. First, you must reindex the source file whenever you modify it, and you can't modify the source file directly from the search program.

The major limitation on modifying a record-compressed text file is that if the modified record is longer than the original, you can't modify it in place. You can overcome this problem by deleting the old record (overwriting it with O's or some other character) and appending the modification to the end of the file.

One disadvantage of this method is that with extensive use the file size quickly becomes too large and you must use a reformatting utility to reorganize the file (eliminating the deleted records). A minor disadvantage is that the position of a record within a file changes every time you modify it.

My solution to this problem (a program called 3by5) is to assign one record per sector. Although this method doesn't optimize disk space, it's simple to implement and doesn't require periodic file maintenance with a reformatting utility.

Another problem with Mindex and Search is that they are designed for systems with disk drives. Random access is slow on disk drives, and Search bypasses this problem by storing as much of the index file in RAM as possible. Large hard disk files can have indexes one or two orders of magnitude greater than available memory.

The solution is simple: 251 random access records are set aside for the index file. The n^{th} bits of all the signature strings are stored in the n^{th} logical record of the file. Only the index records corresponding to the hash values of the search key are read from the disk. You don't need to sequentially read the entire index file.

This file access method, however, requires multiple, nonsequential disk reads from the index file, and a floppy disk based system is too slow. I am in the process of modifying Mindex and Search for TRSDOS, TRS-XENIX, and LDOS hard-disk systems.

Future of Text Storage

Several publishing companies have plans to distribute textual material on laser video discs. These discs hold large amounts of information (text, numerical, program, and pictorial), can be interfaced to microcomputers, and can be mass produced.

The tremendous amount of information stored on the video disc creates a problem. How can you rapidly search large text files (reference works such as encyclopedias, thesauruses, and so on)? Sequential access methods are too slow for these huge files. Tree-searching algorithms aren't appropriate for text.

Signature screening is a partial answer to this problem. You must make substantial modifications to apply this method to extremely large files stored on machines with relatively slow direct access times. The goal is to devise hashing algorithms that maximize the specificity of the screening test (reduce collisions).

The application of signature screening isn't limited to text files. With modification, you can apply it to records with fixed fields. Indexed sequential and signature screening methods can access a data file. You can extend the concept of signature screening in Mindex and Search to include many sophisticated applications. ■

Joseph E. Trojak is the president of Softshell Corporation, P.O. Box 18522, Baltimore, MD 21237. Softshell sells 3by5, an enhanced information retrieval system that incorporates Mindex and Search.

Mindex and Search are available from the author at $12.50 for a TRSDOS 1.3 Model III/4 disk and $14 for a TRSDOS 2.A Model II/12/16 disk. Specify source code or the assembled version. The price includes U.S. postage; Maryland residents must add 5 percent sales tax.

Listing 6 continued

```
        JP      DOSCMD          ;just in case
;---------------------------------------------------------
                                ;ERRDSP
                                ;-------------------------
                                ;error display
ERRDSP  OR      0C0H            ;set bit 6 (detailed error msg) and
                                ;set bit 7 (return to caller)
        CALL    DERDSP          ;call error dsplay message
        RET
;---------------------------------------------------------
DISBRK  RET                     ;for Mod II compatibility
;---------------------------------------------------------
STMEND  LD      DE,(EOMEM)
        LD      (MEMEND),DE     ;store end of memory
        RET                     ;must be called before DE changed!!!!
MEMEND  DEFW    0               ;end of memory storage
;---------------------------------------------------------
                                ;MODIII buffers
                                ;-------------------------
INDCB   DEFB    0               ;input file DCB
IDCB1   DEFB    0
        DEFB    0
        DEFB    0
        DEFB    0
        DEFB    0
        DEFB    0
        DEFB    0
IDCBOS  DEFB    0               ;EOF offset
IDCBRL  DEFB    0               ;logical record length
IDCBCR  DEFW    0               ;current record (actually next record)
IDCBLR  DEFW    0               ;last record
        DEFS    40              ;extents
INLLR   DEFW    0               ;true last sector
;
MAPDCB  DEFB    0               ;MAP file DCB
        DEFB    0
        DEFB    0
        DEFB    0
        DEFB    0
        DEFB    0
        DEFB    0
        DEFB    0
MDCBOS  DEFB    0               ;EOF offset
MDCBRL  DEFB    0               ;logical record length
MDCBCR  DEFW    0               ;current record (actually next record)
MDCBLR  DEFW    0               ;last record
        DEFS    40              ;extents
MAPLLR  DEFW    0               ;true last sector
OUTDCB  DEFS    60              ;output file DCB
SORLN   DEFB    0               ;start of record line number
                                ;for each line of rec displayed
                                ;(SORLN) is decremented
;..........
ACCESS  DEFB    0               ;dummy for Mod II compatability
;---------------------------------------------------------
STMAP   DEFB    0
        END
```

Program Listing 7. Assembly-language listing of S30PT/SRC:1.

```
;S3OPT/SRC.1.2
;Jun 21, 1983
;---------------------------------------------------------
;SEARCH OPTIONS ROUTINE
;COPYRIGHT 1983 SOFTSHELL CORPORATION   ALL RIGHTS RESERVED
;---------------------------------------------------------
S3OPT   PSECT                   ;start of relocatable program section
;---------------------------------------------------------
        PUBLIC  OPT
;---------------------------------------------------------
        EXTERN  KBLINE,KBLIN3,KBCHAR,KBWAIT
        EXTERN  KBBRK
        EXTERN  VDCLS,VDLINE,VDCHAR
        EXTERN  HOME,BOTTOM,NEWLN
        EXTERN  CURPOS,POSCUR
        EXTERN  ERASEL,ERASES
        EXTERN  OPEN,CLOSE,POSEOF,POSN
        EXTERN  OUTDCB
        EXTERN  ARROWD,ARROWL,ARROWR,ARROWU,BRKKEY,SWIDTH,PENULN
;
        EXTERN  NINSET,SINSET,SDMODE,PRMODE,OMODE,ANSWER
        EXTERN  PSEOR,OFSTAT,OFCOND,OPNOUT
        EXTERN  OFNAME,KEYBUF
        EXTERN  APPEND
;---------------------------------------------------------
OPTCOL  EQU     42              ;Column for option answers
;---------------------------------------------------------
                                ;OPT
                                ;-------------------------
                                ;options routine
                                ;entry: no paraeters passed
                                ;exit:  no parameters returned
                                ;       one exit point: OPTEND
                                ;calls: OPTPCB,OPTPC
                                ;reg altered: AF.BC,DE,HL
OPT     LD      BC,4*256        ;r5:cl
        CALL    POSCUR
        CALL    ERASES          ;erase to end of screen
OPT0    LD      A,(PSEOR)       ;get pseudo EOR char
        CP      0DH             ;is it standard EOR (0DH)?
        JR      Z,OPT0R         ;jp OPT0R if record format
        LD      HL,MSGPF        ;'Paragraph Format'03
        CALL    VDLINE
        JR      OPT0E
OPT0R   LD      HL,MSGRF        ;'Record Format'03
        CALL    VDLINE
OPT0E   NOP                     ;end of OPT0 section
;.........................................................
        LD      BC,5*256        ;r6:cl
        CALL    POSCUR
        LD      HL,OPMSG1       ;'Inset from Left Margin'03
        CALL    VDLINE
        LD      A,(NINSET)      ;get inset value
        ADD     A,48            ;convert to ASCII
        CALL    VDCHAR          ;display inset value
;
```

Listing 7 continued

Listing 7 continued

```
         LD      BC,6*256         ;r7:cl
         CALL    POSCUR
         LD      HL,OPMSG2        ;'Search Display Mode: C/R/L'03
         CALL    VDLINE
         LD      A,(SDMODE)       ;get search display mode
         CALL    VDCHAR           ;display search display mode
;
         LD      BC,7*256         ;r8:cl
         CALL    POSCUR
         LD      HL,OPMSG3        ;'Print Mode: C/M'03
         CALL    VDLINE
         LD      A,(PRMODE)       ;get print mode
         CALL    VDCHAR           ;dsp print mode
;
         LD      BC,8*256         ;r9:cl
         CALL    POSCUR
         LD      HL,OPMSG4        ;'Output File: R/A/C'03
         CALL    VDLINE
         LD      HL,OFNAME        ;HL->output file name
         CALL    VDLINE           ;dsp output file name
;
         LD      BC,9*256         ;r10:cl
         CALL    POSCUR
         LD      HL,OPMSG5        ;'Output File: R/A/C'03
         CALL    VDLINE
         LD      A,(OFCOND)       ;get output file status
         CALL    VDCHAR           ;dsp output file status
;
         LD      BC,10*256        ;r11:cl
         CALL    POSCUR
         LD      HL,OPMSG6        ;'Output Mode: C/M'03
         CALL    VDLINE
         LD      A,(OMODE)        ;get output mode
         CALL    VDCHAR           ;dsp output mode
;..................................................................
OPT01    CALL    OPTPCB           ;inset from left margin
         LD      B,5
         CALL    OPTPC            ;r6:c(OPTCOL+1)
         LD      HL,ANSWER
         LD      B,1
         CALL    KBLIN3
         LD      A,0
         CP      B
         JR      Z,OPT01E         ;jp OPT01E if null entry
         LD      A,(ANSWER)
         CP      58
         JP      P,OPT01          ;loop if not no.
         CP      48
         JP      M,OPT01          ;loop if not no.
         SUB     48               ;convert to binary no.
         LD      (NINSET),A       ;save in inset buffer
OPT01E   LD      B,5              ;display inset value
         CALL    OPTPC            ;r6:c(OPTCOL+1)
         LD      A,(NINSET)
         ADD     A,48             ;convert to ASCII
         CALL    VDCHAR           ;dsp inset
;..................................................................
                                  ;OPT02
                                  ;................................
                                  ;search display mode (C/R/L)
OPT02    CALL    OPTPCB           ;r(PENULN):cl
         LD      HL,OPMG2A        ;'(C)ontinuous, (R)ecord or (L)ine'03
         CALL    VDLINE
         LD      B,6
         CALL    OPTPC            ;r7:c(OPTCOL+1)
         LD      HL,ANSWER
         LD      B,1
         CALL    KBLIN3
         LD      A,0
         CP      B
         JR      Z,OPT02E         ;jp OPT02E if null entry
         LD      A,(ANSWER)
         RES     5,A              ;convert to upper case
         CP      'C'
         JP      Z,OPT02D
         CP      'R'
         JP      Z,OPT02D
         CP      'L'
         JR      NZ,OPT02         ;loop until valid input
OPT02D   LD      (SDMODE),A       ;save new search display mode
OPT02E   LD      B,6              ;dsp search display mode
         CALL    OPTPC            ;r7:c(OPTCOL+1)
         LD      A,(SDMODE)
         CALL    VDCHAR
;..................................................................
                                  ;OPT03
                                  ;................................
                                  ;print mode (C/M)
OPT03    CALL    OPTPCB           ;r(PENULN):cl
         LD      HL,OPMG3A        ;'(C)ontinuous or (M)anual'03
         CALL    VDLINE
         LD      B,7
         CALL    OPTPC            ;r8:C(OPTCOL+1)
         LD      HL,ANSWER
         LD      B,1
         CALL    KBLIN3
         LD      A,0
         CP      B
         JR      Z,OPT03E         ;jp OPT03E if null entry
         LD      A,(ANSWER)
         RES     5,A              ;convert to upper case
         CP      'C'
         JP      Z,OPT03D
         CP      'M'
         JR      NZ,OPT03         ;loop until valid answer
OPT03D   LD      (PRMODE),A       ;store new print mode
OPT03E   LD      B,7
         CALL    OPTPC            ;r8:c(OPTCOL+1)
         LD      A,(PRMODE)
         CALL    VDCHAR           ;display print mode
;..................................................................
                                  ;OPT04
                                  ;................................
                                  ;output file name
OPT04    CALL    OPTPCB           ;r(PENULN):cl
OPT04A   LD      B,8
         LD      C,42
         CALL    POSCUR           ;r9:c43
         CALL    ERASEL           ;erase to end of line
         LD      HL,OFNAME        ;temp store file name in KEYBUB
         LD      DE,KEYBUF
OPT04B   CALL    APPEND           ;move OFNAME to KEYBUF
         LD      HL,OFNAME
         LD      B,24
         CALL    KBLIN3           ;get new file name
         DEC     HL
         LD      (HL),0DH         ;0DH needed for file open
         LD      A,0
         CP      B
         JP      Z,OPT04N         ;jp OPT04N if null entry
         LD      DE,OUTDCB        ;DE->output file DCB
         CALL    CLOSE            ;close output file
         LD      A,'C'
         LD      (OFSTAT),A       ;output file status = (C)losed
         LD      HL,OFNAME        ;HL->output file name
         LD      DE,OUTDCB        ;DE->output file DCB
OPT04L   CALL    APPEND           ;move output file name to OUTDCB
         CALL    OPNOUT           ;open output file
         LD      A,(OFSTAT)
         CP      'C'
         JP      Z,OPT04A
         JP      OPT04E
;
OPT04N   LD      HL,KEYBUF        ;default name
         LD      DE,OFNAME
OPT04Q   CALL    APPEND           ;move output file name from KEYBUF to OFNAME
         LD      (HL),03          ;avoid new line feed
         LD      B,8
         LD      C,42
         CALL    POSCUR           ;r9:c43
```

Listing 7 continued

Listing 7 continued

```
        LD      HL,KEYBUF
        CALL    VDLINE          ;display output file name
        LD      A,(OFSTAT)
        CP      'C'
        JP      NZ,OPT04E
        LD      HL,OFNAME
        LD      DE,OUTDCB
        CALL    APPEND          ;move file spec to OUTDCB
OPT04E  NOP
;...............................................................
                                ;OPT05
                                ;...............................
                                ;output file status (R/A/C)
OPT05   CALL    OPTPCB          ;r(PENULN):cl
        LD      HL,OPMG5A       ;'(R)eset, (A)ppend or (C)lose'03
        CALL    VDLINE
        LD      B,9
        CALL    OPTPC           ;r10:c(OPTCOL+1)
        LD      HL,ANSWER
        LD      B,1
        CALL    KBLIN3
        LD      A,0
        CP      B
        JP      Z,OPT05E        ;jp OPT05E if null entry
        LD      A,(ANSWER)
        RES     5,A             ; convert to upper case
        LD      (ANSWER),A
        CP      'C'
        JP      Z,OPT05C
        CP      'R'
        JP      Z,OPT05R
        CP      'A'
        JP      Z,OPT05A
        JP      OPT05           ;loop until valid entry
OPT05C  LD      A,(OFSTAT)
        CP      'C'
        JP      Z,OPT05D        ;file already closed
        LD      DE,OUTDCB       ;DE->output file DBC
        CALL    CLOSE           ;close output file
        JP      OPT05D
;...............................................................
                                ;OPT05R
                                ;...............................
                                ;open output file and reset
                                ;file cursor to start of file
OPT05R  CALL    OPNOUT          ;open output file
        LD      A,(OFSTAT)
        CP      'C'
        JP      Z,OPT05         ;loop if error in opening file
        LD      DE,OUTDCB       ;DE->output file DCB
        LD      BC,0
        CALL    POSN            ;position file cursor to first record
        JP      OPT05D
;...............................................................
                                ;OPT05A
                                ;...............................
                                ;open output file and set file
                                ;file cursor to EOF
OPT05A  CALL    OPNOUT          ;open output file
        LD      A,(OFSTAT)
        CP      'C'
        JP      Z,OPT05         ;loop if error in opening file
        LD      DE,OUTDCB
        CALL    POSEOF          ;position to end of file
;...............................................................
                                ;OPT05D
                                ;...............................
                                ;end of OPT05 section
OPT05D  LD      A,(ANSWER)
        LD      (OFCOND),A
OPT05E  LD      B,9
        CALL    OPTPC           ;r10:c(OPTCOL+1)
        LD      A,(OFCOND)
        CALL    VDCHAR          ;display output file status
;...............................................................
                                ;OPT06
                                ;...............................
                                ;output mode (C/M)
OPT06   CALL    OPTPCB          ;r(PENULN):cl
        LD      HL,OPMG3A       ;'(C)ontinuous or (M)anual'03
        CALL    VDLINE
        LD      B,10
        CALL    OPTPC           ;r11:c(OPTCOL+1)
        LD      HL,ANSWER
        LD      B,1
        CALL    KBLIN3
        LD      A,0
        CP      B
        JP      Z,OPT06E
        LD      A,(ANSWER)
        RES     5,A             ;convert to upper case
        CP      'C'
        JP      Z,OPT06D
        CP      'M'
        JP      NZ.OPT06
OPT06D  LD      (OMODE),A
OPT06E  LD      B,10            ;dsp output mode
        CALL    OPTPC           ;r11:c(OPTCOL+1)
        LD      A,(OMODE)
        CALL    VDCHAR
        LD      BC.3*256        ;r4:cl
        CALL    POSCUR
        CALL    ERASES          ;erase to end of screen
OPTEND  RET                     ;end of options
;---------------------------------------------------------------
                                ;OPTPCB
                                ;-------------------------------
                                ;pos cursor to next to bottom line
                                ;r23:cl Mod II, r15:cl Mod III
OPTPCB  LD      B,PENULN        ;next to bottom line
        LD      C,0             ;col 1
        JR      OPTPC3          ;jp to position section
;---------------------------------------------------------------
                                ;OPTPC
                                ;-------------------------------
                                ;position cursor to col = OPTCOL
                                ;B=line number
OPTPC   LD      C,OPTCOL        ;B passed as parameter
OPTPC3  CALL    POSCUR
        CALL    ERASEL          ;erase to end of line
        RET
;---------------------------------------------------------------
                                ;OPT MESSAGES
                                ;-------------------------------
OPMSG1  DEFM    'Inset from Left Margin        '
        DEFB    03
OPMSG2  DEFM    'Search Display Mode: C/R/L    '
        DEFB    03
OPMSG3  DEFM    'Print Mode: C/M               '
        DEFB    03
OPMSG4  DEFM    'Output File Name: '
        DEFB    03
OPMSG5  DEFM    'Output File: R/A/C            '
        DEFB    03
OPMSG6  DEFM    'Output Mode: C/M              '
        DEFB    03
OPMG2A  DEFM    '(C)ontinuous. (R)ecord or (L)ine'
        DEFB    03
OPMG3A  DEFM    '(C)ontinuous or (M)anual'
        DEFB    03
OPMG5A  DEFM    '(R)eset, (A)ppend or (C)lose'
        DEFB    03
MSGPF   DEFM    'Paragraph Format'
        DEFB    03
MSGRF   DEFM    'Record Format'
        DEFB    03
;---------------------------------------------------------------
        END
```

Program Listing 8. Assembly-language listing of M3/SRC.

```
;M3/SRC  VERSION 1.2
;Jun 26, 1983
;---------------------------------------------------------------
;MINDEX INDEXING PROGRAM
;MAIN MODULE
;COPYRIGHT 1983, SOFTSHELL CORPORATION   ALL RIGHTS RESERVED
;---------------------------------------------------------------
MINDEX  PSECT                   ;begin relocatable program section
;---------------------------------------------------------------
                                ;public variables
                                ;-------------------------------
        PUBLIC  ANSWER          ;1 byte switch
        PUBLIC  BEGIN           ;start of MINDEX program
        PUBLIC  CMDLFS          ;cmd line file spec (Y/N)
                                ;indicates whether file spec also entered
                                ;along with MINDEX cmd
        PUBLIC  KEYBUF          ;keybuf used to enter MINDEX file spec
;---------------------------------------------------------------
                                ;calls to MODII or MODIII
                                ;-------------------------------
        EXTERN  BOTTOM          ;position cursor to last line of screen
        EXTERN  CLOSE           ;close file (DE->DCB)
        EXTERN  CURPOS          ;determine cursor pos (coor:BC)
        EXTERN  DISBRK          ;disable break (for Model II)
        EXTERN  DIVIDE          ;divide A into HL
        EXTERN  DOSCMD          ;jp to TRSDOS and execute command
        EXTERN  ERASEL          ;erase to end of line
        EXTERN  ERASES          ;erase to end of screen
        EXTERN  ERRDSP          ;MAP error display routine
        EXTERN  ERROR           ;TRSDOS error msg routine
        EXTERN  HOME            ;position cursor to top line of screen
        EXTERN  INIT            ;initialize file (create/open, pos to 1st rec)
                                ;                (DE->DCB, HL->rec buf)
        EXTERN  JP2DOS          ;jp to TRSDOS
        EXTERN  KBCHAR          ;get char (or null) from keyboard
        EXTERN  KBLINE          ;display line pointed to by HL
        EXTERN  KBWAIT          ;wait for char from keyboard
        EXTERN  OPEN            ;open file (DE->DCB, HL->record buffer)
        EXTERN  POSCUR          ;position cursor (coordinates in BC)
        EXTERN  READDR          ;read file: direct access
        EXTERN  READNX          ;read next record in file
        EXTERN  VDCHAR          ;display char in A register
        EXTERN  VDCLS           ;clear screen
        EXTERN  VDINIT          ;initialize video I/O (for Model II)
        EXTERN  VDLINE          ;display line (HL->msg)
        EXTERN  WRITNX          ;write next record to file
;...............................................................
                                ;MODII/III buffers and I/O parameters
                                ;...............................
        EXTERN  INDCB           ;INput (source) file DCB
        EXTERN  IDCBRL          ;Input file DCB Record Length
        EXTERN  IDCBCR          ;Input file DCB Current Record
        EXTERN  IDCBLR          ;Input file DCB Last Record
        EXTERN  INLLR           ;true last record of source file
        EXTERN  IDCBOS          ;Input file DCB xxxxxxx
        EXTERN  MAPDCB          ;MAP (index) file DCB
        EXTERN  MDCBRL          ;Map file DCB Record Length
        EXTERN  MDCBCR          ;Map file DCB Current Record
        EXTERN  MDCBLR          ;Map file DCB Last Record
        EXTERN  MAPLLR          ;true last record of index file
        EXTERN  MDCBOS          ;Map file DCB xxxxxxxx
                                ;I/O parameters:
        EXTERN  BRKKEY          ;break key value
        EXTERN  CPYPOS          ;copyright notice position (52 or 36)
        EXTERN  PENULN          ;next to last line on screen (23 or 15)
        EXTERN  SWIDTH          ;screen width (80 or 64)
;
;---------------------------------------------------------------
                                ;calls to GR (general routines)
                                ;-------------------------------
        EXTERN  BXDEC           ;binary to decimal conversion
        EXTERN  DECXB           ;decimal to binary conversion
        EXTERN  DLINE           ;display dotted line (e.g. ...........)
        EXTERN  DSPNBR          ;display 5-digit decimal no. (right justified)
        EXTERN  DSPNBRL         ;display 5-digit decimal no. (left justified)
        EXTERN  EATBNK          ;remove leading blanks
        EXTERN  FSPEC           ;move cmd line file psec to KEYBUF
        EXTERN  GETMFN          ;get MAP file names
        EXTERN  LINE            ;display line (from char in A reg)
        EXTERN  PACKFS          ;pack (compress) file spec in fixed table
        EXTERN  PARSEF          ;parse one file spec
        EXTERN  REC0            ;zero-out 256-byte record
        EXTERN  RECBL           ;blank-out 256-byte record
        EXTERN  TAPENT          ;'Tap enter to continue'
;---------------------------------------------------------------
                                ;GR buffers
                                ;-------------------------------
        EXTERN  SNAM,SEXT,SPSW,SDDR     ;source file table
        EXTERN  INAM,IEXT,IPSW,IDDR     ;index file table
;
;---------------------------------------------------------------
;---------------------------------------------------------------
                                ;MINDEX MAIN SEGMENT
                                ;syntax for loading program:
                                ;MINDEX source file,index file;options
                                ;options:
                                ; R=record format or P=paragraph format
                                ; T=return to TRSDOS or M return to MAP
                                ; R and M are default values
;---------------------------------------------------------------
BEGIN   PUSH    HL              ;HL->cmd line buffer
                                ;save while break is disabled
        CALL    DISBRK          ;disable break on Model II
        POP     HL              ;HL->cmd line buffer
        CALL    FSPEC           ;get cmd line file spec if any
        CALL    VDCLS           ;clear screen
        LD      HL,TITLE        ;point to title
        CALL    VDLINE          ;display title
        LD      BC,CPYPOS       ;copyright notice coordinates (Mod II/III)
        CALL    POSCUR          ;r1:c  Model II,  r1:c   Model III
        LD      HL,CPYMSG       ;point to copyright notice
        CALL    VDLINE          ;display copyright notice
        LD      A,'-'           ;select '-' char for LINE routine
        CALL    LINE            ;draw dashed line under title
;---------------------------------------------------------------
                                ;FN01
                                ;-------------------------------
FN01    LD      A,'R'           ;set default values
        LD      (FORMAT),A      ;deafult indexing format = record
        LD      A,'M'
        LD      (ENDCMD),A      ;default = return to MAP when done
        LD      BC,3*256
        CALL    POSCUR          ;r4:cl
        CALL    ERASES          ;erase to end of screen
        LD      HL,FNMSG        ;point to file name msg ('File Name:')
        CALL    VDLINE          ;display file name message
        LD      B,PENULN        ;point to next to last line
        LD      C,0
        CALL    POSCUR          ;r23:cl Mod II, r15:cl Mod III
        LD      HL,FNMSG2       ;point to following message:
                                ;'Enter name of File to be indexed or
```

Listing 8 continued

Continued on page 211

Listing 8 continued

```
                                        ;? for help'
        CALL    VDLINE                  ;display message
;
        LD      A,(CMDLFS)              ;CMDLFS indicates whether file spec
                                        ;was entered after MINDEX on TRSDOS cmd line
        CP      'Y'                     ;was file spec entered on TRSDOS cmd line?
        JR      NZ,FN02                 ;jp if file name not entered
;.......
                                        ;file spec entered on cmd line so dsp
        LD      BC,3*256+11
        CALL    POSCUR                  ;r4:cl2
        LD      A,'N'                   ;reset CMDLFS
        LD      (CMDLFS),A
        LD      HL,KEYBUF               ;point to cmd line file spec
        CALL    VDLINE                  ;display file spec
        LD      A,0DH
        CP      A                       ;Z (set)
        LD      B,1                     ;(so len > 0)
        JR      FN04
;-----------------------------------------------------------------
FN02    LD      BC,3*256+11             ;get file spec
        CALL    POSCUR                  ;r4:cl2
        CALL    ERASEL                  ;erase to end of line
        LD      HL,KEYBUF               ;point to KEYBUF to hold file spec
        LD      B,51                    ;max file spec = 51 characters
        CALL    KBLINE                  ;enter file spec
;-----------------------------------------------------------------
FN04    PUSH    AF                      ;now process file spec
        PUSH    HL
        LD      BC,5*256
        CALL    POSCUR                  ;r6:cl
        CALL    ERASES                  ;erase to end of screen
        POP     HL
        POP     AF
;
        CP      BRKKEY                  ;was <break> entered?
        JP      Z,MMENU1                ;if so jp to mini-menu 1
        LD      A,(KEYBUF)              ;get first char entered
        CP      0DH                     ;was first char 0DH (i.e. null entry)?
        JR      Z,FN01                  ;loop if null entry
        LD      HL,KEYBUF               ;point to file spec buffer
        CALL    EATBNK                  ;bypass leading blanks
        LD      A,(HL)                  ;load A with first non-blank char
        CP      '?'                     ;was '?' entered?
        JP      Z,MHELP                 ;if so jp to help screen
;
        LD      HL,KEYBUF               ;point to file spec buffer
        CALL    GETMFN                  ;get file names
        JR      Z,FNOK                  ;jp to FNOK if file spec syntax OK
        LD      BC,8*256                ;file spec syntax incorrect
        CALL    POSCUR                  ;r9:c0
        LD      HL,FNEMSG               ;point to file name error msg
        CALL    VDLINE                  ;display file name error msg
        CALL    TAPENT                  ;'Tap enter key to continue'
        JP      FN02                    ;loop back and get file spec again
;-----------------------------------------------------------------
FNOK    NOP                             ;file names ok
        CALL    CIOPT                   ;call command interpreter options routine
        JP      NZ,OPTERR               ;jp to options error routine if error
;
        LD      HL,SNAM                 ;HL->source file name
        LD      DE,INDCB                ;DE->source flie DCB
        CALL    PACKFS                  ;space compress source file spec
                                        ;and move to source file DCB
;
        LD      HL,INAM                 ;HL-> index file table
        LD      DE,MAPDCB               ;DE-> index file DCB
        CALL    PACKFS                  ;space compress index file spec
                                        ;and move to index file DCB
;
        LD      BC,7*256
        CALL    POSCUR                  ;r8:c0
        LD      HL,MSG09                ;pt to msg 'source file name:'
        CALL    VDLINE                  ;dsp msg
        LD      HL,INDCB                ;HL->source file DCB
        CALL    VDLINE                  ;display source file spec
;
        LD      BC,8*256                ;r9:cl
        CALL    POSCUR
        LD      HL,MSG10                ;'map index file name:'
        CALL    VDLINE                  ;dsp above msg
        LD      HL,MAPDCB               ;HL->index file DBC
        CALL    VDLINE                  ;display index file spec
        JP      OPFILE                  ;jp to section to open files
;-----------------------------------------------------------------
OPTERR  LD      HL,OPEMSG               ;HL->options error message
        CALL    VDLINE                  ;dsp options error msg
        CALL    TAPENT                  ;Tap enter key to continue
        JP      FN01                    ;loop back and get file spec
;-----------------------------------------------------------------
                                        ;OPFILE
                                        ;-------------------------------
OPFILE  NOP                             ;open files
        LD      BC,11*256               ;rl2:cl
        CALL    POSCUR
CKFNAM  LD      HL,INDCB                ;HL->source file DCB
        LD      DE,MAPDCB               ;DE->index file DCB
CFNLP   LD      A,(DE)                  ;check if file spec's different
        CP      (HL)
        JR      NZ,DIFFIL               ;not equal so dif names
        CP      0DH
        JR      Z,SAMFIL                ;0DH => same file names
        INC     HL
        INC     DE
        JR      CFNLP
SAMFIL  LD      HL,MSGSFN               ;same file names error msg
        CALL    VDLINE                  ;dsp error msg
        CALL    TAPENT                  ;Tap enter to continue
        JP      FN01                    ;loop back and get file spec again
;-----------------------------------------------------------------
                                        ;DIFFIL
                                        ;-------------------------------
                                        ;different file spec for source and index
                                        ;now print msg regarding indexing format
DIFFIL  LD      A,(FORMAT)              ;paragraph or record format
        CP      'P'                     ;paragraph format?
        CALL    Z,PERIOD                ;paragraph format so set proagram for '.'
        LD      BC,9*256                ;rl0:cl
        CALL    POSCUR
        LD      A,(FORMAT)
        CP      'P'                     ;paragraph format?
        JP      Z,FTDSP4                ;if para format jp to appropriate dsp
        LD      HL,RECMSG               ;pt to record format message
        CALL    VDLINE                  ;display record format msg
        JP      FTDSP6                  ;bypass paragraph format msg
FTDSP4  LD      HL,PARMSG               ;pt to paragraph format msg
        CALL    VDLINE                  ;display paragraph format msg
FTDSP6  NOP
;-----------------------------------------------------------------
PREIN   NOP                             ;prepare input (source) file
        LD      A,0DH
        CALL    VDCHAR                  ;skip to next line
        LD      HL,INBUF                ;HL->input buffer
        LD      DE,INDCB                ;DE->source file DCB
        LD      B,0                     ;recrod length = 256
        CALL    OPEN                    ;open source file
        JP      Z,PREIN8                ;jp to PREIN8 if open OK
        PUSH    AF                      ;save error code
        LD      BC,12*256               ;rl3:cl
        CALL    POSCUR
        LD      HL,MSGSER               ;source file open error msg
        CALL    VDLINE                  ;dsp source file open error msg
        LD      BC,13*256               ;rl4:cl
        CALL    POSCUR
        POP     AF                      ;restore error code
        CALL    ERRDSP                  ;display TRSDOS error msg
        LD      BC,14*256               ;rl5:cl
        CALL    POSCUR
        CALL    TAPENT                  ;Tap enter to continue
        JP      FN01                    ;loop and get file spec again
PREIN8  LD      A,0
        LD      (IDCBRL),A              ;set record len = 256
        LD      BC,7*256+37             ;r8:c38
        CALL    POSCUR
        LD      HL,OOKMSG               ;'Opened' msg
        CALL    VDLINE
;-----------------------------------------------------------------
                                        ;PREMAP
                                        ;-------------------------------
                                        ;prepare MAP index file
PREMAP  LD      HL,MAPBUF               ;HL->index file buffer
        LD      DE,MAPDCB               ;DE->index file DCB
        LD      B,0                     ;record length = 256
        CALL    INIT                    ;intialize (create or open) file
        JP      Z,CMSGOP                ;jp to CMSGOP if index file init OK
        PUSH    AF                      ;save error code
        LD      BC,12*256               ;rl3:cl
        CALL    POSCUR
        LD      HL,MSGMER               ;'Error in opening index file'
        CALL    VDLINE                  ;dsp error msg
        LD      BC,13*256               ;rl4:cl
        CALL    POSCUR
        POP     AF                      ;restore error code
        CALL    ERRDSP                  ;dsp TRSDOS error msg
        LD      BC,14*256               ;rl5:cl
        CALL    POSCUR
        CALL    TAPENT                  ;Tap enter key to continue
        JP      FN01                    ;loop back and get file spec
CMSGOP  LD      BC,8*256+37             ;index file open OK
        CALL    POSCUR                  ;r9:C38
        LD      HL,OOKMSG               ;'Opened' msg
        CALL    VDLINE
;-----------------------------------------------------------------
                                        ;add header to MAP index file
                                        ;header format:
                                        ; bytes 0-19: MINDEX1.2..
                                        ; bytes 40-41: file spec.
                                        ;-------------------------------
        LD      A,0DH
        LD      (SDDR),A                ;set source file drive no. to 0DH (null)
                                        ;since header never specifies disk drive
                                        ;of source file
        LD      HL,SNAM                 ;HL->source file spec table
        LD      DE,HFN                  ;DE->header file name buffer
        CALL    PACKFS                  ;pack header with source file spec
                                        ;space compress file spec in process
;
        LD      HL,MAPBUF               ;HL->index file buffer
        CALL    REC0                    ;zero out index file buffer
        LD      HL,HSEC                 ;header sector
        LD      DE,MAPBUF
        LD      BC,50
        LDIR                            ;move header to MAPBUF
        LD      DE,MAPDCB
        CALL    WRITNX                  ;write index header sector (sector 0)
        JP      NZ,ERROR                ;jp to erro routine if error
        LD      HL,MAPBUF               ;HL->index buffer
        CALL    REC0                    ;initialize index buffer to 0's
;-----------------------------------------------------------------
        LD      BC,256*10               ;rll:cl
        CALL    POSCUR
        LD      HL,IDXSEC               ;'Indexing Sector:'
        CALL    VDLINE                  ;dsp msg
;-----------------------------------------------------------------
                                        ;files open and ready to
                                        ;start indexing process
;-----------------------------------------------------------------
                                        ;main loop, repeat until file indexed
LOOP    CALL    READIN                  ;get next sector from source file
        LD      A,(RDERR)               ;RDERR = error code during read
        CP      28                      ;end of file?
        JP      Z,ENDIN                 ;jp to EDNIN if end of source file
                                        ;this is the only EXIT from this loop
        LD      C,255                   ;IO buffer length
        LD      A,(SCROSS)              ;no. of bytes in current record
                                        ;carried over from previous sector
                                        ;if SCROSS=0 then record did not span sector
                                        ;SCROSS never > 255
        LD      B,A                     ;B now has number of bytes in record
                                        ;from previous sector
        CALL    HASHWD                  ;hash all the words in current sector
                                        ;on return from HASHWD IX->byte after
                                        ;source file record buffer
        DEC     IX                      ;IX now -> last byte in src file rec buf
        LD      A,(IX+0)                ;get last char in SFRB
POINT1  CP      0DH                     ;'.' ?
                                        ;if file indexed in paragraph format,
                                        ;the above 0DH is converted to '.'
        JP      Z,NCROSS                ;jp if record does not span sector
        CP      0DH                     ;EOR ?
        JP      Z,NCROSS                ;jp if record does not span sector
        LD      A,B
        LD      (SCROSS),A              ;store in SCROSS no. of bytes in record
                                        ;before sector boundary crossed
        JP      LOOP                    ;loop until end of source file
NCROSS  LD      A,0
        LD      (SCROSS),A              ;SCROSS=0 means record does not
                                        ;span sector boundary
                                        ;0DH at end of source file rec buf
        CALL    NEWSEC                  ;increment BICT
        JP      LOOP                    ;loop until end of source file
;-----------------------------------------------------------------
;-----------------------------------------------------------------
                                        ;HASHWD
                                        ;-------------------------------
                                        ;hash word subroutine
                                        ;hashes all valid triplets in record
                                        ;hash algorithm for three chars cl,c2,c3
                                        ;h(cl,c2,c3)=
                                        ;  (100acl+10ac2+ac3) mod 251
                                        ;where: cl,c2,c3 represent the lst,2nd and
                                        ;         3rd char of the triplet
                                        ;       a is the ASCII value of the char c
                                        ;          and upper case is converted to l.c.
                                        ;three 2-byte buffers are used:
                                        ;FBYTE=100acl
                                        ;MBYTE=10ac2
                                        ;LBYTE=ac3
                                        ;
                                        ;during hash loop reg B is used to store
                                        ;no. of bytes in current logical record
                                        ;value in B is transferred to SCROSS
                                        ;only when a record spans a sector
                                        ;and a new sector must be read from the
                                        ;source file
                                        ;
                                        ;during hash loop reg C is used to store
                                        ;no. of bytes remaining if sfrb
;
HASHWD  LD      IX,INBUF                ;IX->source file  record buffer (sfrb)
BUFLP   LD      HL,(MBYTE)              ;move MBYTE to HL
        CALL    MUHL10                  ;multiply MBYTE by 10
        LD      (FBYTE),HL              ;move HL to FBYTE
;
        LD      HL,(LBYTE)              ;move LBYTE to HL
        CALL    MUHL10                  ;multiply HL by 10
        LD      (MBYTE),HL              ;move HL to MBYTE
;
        LD      A,(IX+0)                ;get c3 from sfrb
;...............
POINT2  CP      0DH                     ;'.' ?
                                        ;if file is indexed in paragraph format,
                                        ;the above 0DH is converted to '.'
        JP      Z,ZERO                  ;if char = EOR, then A = 0
        CP      91
        JP      P,LCASE                 ;jp if lower case - no conversion needed
        CP      65
        JP      P,UPCASE                ;jp to UPCASE to convert u.c. to l.c.
        CP      33
        JP      M,ZERO                  ;jp if value <33 (i.e. control or ' ')
        JR      LCASE                   ;char with ASCII value from 33 to 64
```

Listing 8 continued

Listing 8 continued

```
                                    ;do not convert to l.c.
UPCASE  OR      32                  ;convert u.c. to l.c.
        JR      LCASE
ZERO    LD      A,0                 ;c3 not a valid triplet char
LCASE   LD      L,A
        LD      H,0                 ;load HL with value of character
;-----------------------------------------------------------------
;.............................start sector cross routine
        LD      A,(SCROSS)
        CP      0                   ;check SCROSS
        JP      Z,BUFLP3            ;jp BUFLP3 if SCROSS=0
;........................
        LD      A,(IX+0)            ;get c3 from sfrb again
POINT3  CP      0DH                 ;'.' ?
                                    ;if file indexed in paragraph format,
                                    ;convert above 0DH to '.'
        JP      Z,BUFLP2
        CP      0DH                 ;EOR ?
        JP      NZ,BUFLP5
                                    ;EOR while SCROSS = 1
BUFLP2  CALL    NEWSEC              ;shift sector ct
        LD      A,0
        LD      (SCROSS),A          ;reset SCROSS = 0
        LD      B,0
        JP      BUFLP5
;........................
BUFLP3  LD      A,(IX+0)            ;get c3 from sfrb again
POINT4  CP      0DH                 ;'.' ?
                                    ;if file indexed in paragraph format,
                                    ;convert above 0DH to '.'
        JP      Z,BUFL3B
        CP      0DH                 ;EOR ?
        JP      NZ,BUFLP4
                                    ;EOR while SCROSS = 0
BUFL3B  LD      B,0
        JP      BUFLP7
BUFLP4  LD      A,254
        CP      B                   ;check rec len
        JP      NZ,BUFLP7           ;JP BUFLP7 if <255
        LD      B,0                 ;set record len = 0
        JP      BUFLP8              ;SCROSS=0 so no need to
                                    ;call NEWSEC routine
;........................
BUFLP5  LD      A,254
        CP      B                   ;check for rec >255 bytes
        JP      NZ,BUFLP7
                                    ;REC >255 bytes
        CALL    NEWSEC              ;bump sector ct
        LD      B,0                 ;reset rec len = 0
        LD      A,B
        LD      (SCROSS),A          ;reset SCROSS = 0
        JP      BUFLP8
BUFLP7  INC     B                   ;increment record len
BUFLP8  INC     IX                  ;IX now -> next byte
;........................end 11/17/82 insert C
        LD      (LBYTE),HL
        CALL    CHECK0              ;check triplet for 0 val
                                    ;on retrun A=0 if any
                                    ;char in triplet has 0 value
        CP      0
        JP      Z,CHECKC            ;if 0 do not hash triplet
        LD      HL,(LBYTE)          ;HL=100ac1
        LD      DE,(MBYTE)          ;DE=10ac2
        ADD     HL,DE               ;100ac1+10ac2
        LD      DE,(FBYTE)          ;DE=ac3
        ADD     HL,DE               ;(100ac1+10ac2)+ac3
        LD      A,251               ;divisor
        CALL    DIVIDE              ;A has modulo 251 result
        CALL    STORE               ;set bit in appro. sig. sector byte
;
CHECKC  LD      A,0                 ;check if end of record reached
        CP      C
        RET     Z                   ;return if end of record
        DEC     C
        JP      BUFLP               ;continue to hash record until
                                    ;end of record (i.e. C=0)
                                    ;BUFLP end
;-----------------------------------------------------------------
MUA10   NOP                         ;mult # in A by 10
                                    ;BC and DE used
                                    ;value returned in HL
        LD      H,0
        LD      L,A
MUHL10  NOP                         ;multiply # in HL by 10
        ADD     HL,HL               ;HL contains 2*no.
        LD      E,L
        LD      D,H
        ADD     HL,HL               ;HL contains 4*no.
        ADD     HL,HL               ;HL contains 8*no.
        ADD     HL,DE               ;HL contains 10*no.
        RET
;-----------------------------------------------------------------
                                    ;STORE
                                    ;---------------------------------
                                    ;store bit in appro byte
                                    ;of output buffer
STORE   LD      IY,MAPBUF           ;set IY to point to index file buffer
        LD      E,A                 ;move hash value to DE
        LD      D,0
        ADD     IY,DE               ;add hash value (DE) to index file buffer
        LD      A,(BICT)            ;sector pointer
        OR      (IY+0)
        LD      (IY+0),A
        RET
;-----------------------------------------------------------------
                                    ;NEWSEC
                                    ;---------------------------------
                                    ;new source file sector so
                                    ;rotate BICT bit
NEWSEC  LD      A,(BICT)
        RLCA
        LD      (BICT),A
        CALL    C,WRITER            ;write rec if new sector
        RET
;-----------------------------------------------------------------
                                    ;CHECK0
                                    ;---------------------------------
                                    ;check triplet for 0's
                                    ;if any char in triplet has 0 value,
                                    ;then triplet is not hashed
                                    ;blanks, EOR, control char give 0 value
                                    ;on return A=0 -> 0 value
CHECK0  LD      DE,0
        OR      A                   ;carry = 0
        LD      HL,(LBYTE)
        SBC     HL,DE
        JP      Z,VAL0
        LD      HL,(MBYTE)
        OR      A                   ;carry = 0
        SBC     HL,DE
        JP      Z,VAL0
        LD      HL,(FBYTE)
        OR      A                   ;carry = 0
        SBC     HL,DE
        JP      Z,VAL0
        LD      A,1
        RET
VAL0    LD      A,0                 ;zero value for triplet, do not hash
        RET
;-----------------------------------------------------------------
READIN  NOP                         ;read input record
        LD      DE,INDCB            ;DE->source file DCB
        CALL    READNX              ;read next sector
        JP      Z,READOK            ;jp if read OK
        CP      28                  ;end of file?
        JP      NZ,ERROR            ;jp to error routine if not EOR
        LD      (RDERR),A           ;store EOF code in RDERR
READOK  NOP
;-----------------------------------------------------------------
                                    ;DSPSNR
```

Listing 8 continued

Listing 8 continued

```
                                  ;--------------------------------
                                  ;display current source file sector no.
DSPSNR  PUSH    BC
        PUSH    DE
        PUSH    HL
        PUSH    IX
        PUSH    IY
        LD      BC,10*256+18      ;r11:c18
        CALL    POSCUR
        LD      HL,(IDCBCR)       ;load current source sector no. into HL
        CALL    BXDEC             ;convert binary to ASCII
                                  ;on return HL->buf with 5 digit ASCII value
        CALL    VDLINE            ;display ASCII value
        POP     IY
        POP     IX
        POP     HL
        POP     DE
        POP     BC
        RET
;-----------------------------------------------------------------------
                                  ;PERIOD
                                  ;--------------------------------
                                  ;sets '.' for paragraph format
                                  ;0DH is replaced with '.' to indicate
                                  ;end of record
PERIOD  LD      HL,POINT1
        INC     HL                ;bypass CP instruction
        LD      (HL),'.'          ;convert 0DH to '.'
        LD      HL,POINT2
        INC     HL                ;bypass CP instruction
        LD      (HL),'.'          ;convert 0DH to '.'
        LD      HL,POINT3
        INC     HL                ;bypass CP instruction
        LD      (HL),'.'          ;convert 0DH to '.'
        LD      HL,POINT4
        INC     HL                ;bypass CP instruction
        LD      (HL),'.'          ;convert 0DH to '.'
        LD      HL,HSECPF
        LD      (HL),2EH          ;mark header at byte 12
        RET
;-----------------------------------------------------------------------
                                  ;WRITER
                                  ;--------------------------------
                                  ;write output rec
WRITER  LD      DE,MAPDCB         ;DE->index file DCB
        PUSH    BC                ;save BC
        CALL    WRITNX            ;write next sector
        JP      NZ,ERROR          ;jp if error
        LD      HL,MAPBUF         ;pt to index file buffer
        CALL    REC0              ;set index file buffer to 0's
        POP     BC                ;restore BC
        RET
;-----------------------------------------------------------------------
                                  ;CIOPT
                                  ;--------------------------------
                                  ;command interpreter options routine
                                  ;HL -> first NB char after file spec
CIOPT   CALL    EATBNK            ;bypass blanks before semicolon
        LD      A,(HL)
        CP      ';'               ;semicolon entered to mark options?
        JR      NZ,CPT2           ;jp if no options
        INC     HL                ;point to next char
        CALL    EATBNK            ;bypass any blanks after semicolon
CPT2    CALL    EATBNK            ;remove any leading blanks
        LD      A,(HL)
        CP      0DH               ;? EOR
        JR      Z,CPTOK
        CP      ','
        JR      Z,CPT4
        RES     5,A               ;convert to upper case
        CP      'R'
        JR      Z,CPTFMT
        CP      'P'
        JR      Z,CPTFMT
        CP      'T'
        JR      Z,CPTECM
        CP      'M'
        JR      Z,CPTECM
        JP      CPTBAD
CPT4    INC     HL
        JR      Z.CPT2            ;loop and continue to process options
                                  ;until end of cmd line reached
;
CPTFMT  LD      (FORMAT),A        ;load format parameter
        JR      CPT4
CPTECM  LD      (ENDCMD),A        ;load end command parameter
        JR      CPT4
;
CPTBAD  LD      A,0               ;bad options syntax
        CP      1                 ;NZ (reset)
        RET
CPTOK   CP      A                 ;Z (set)  options syntax OK
        RET
;-----------------------------------------------------------------------
                                  ;MMENU1
                                  ;--------------------------------
MMENU1  NOP                       ;mini-menu 1
                                  ;used to abort program before indexing started
        LD      BC,2*256          ;r3:cl
        CALL    POSCUR
        CALL    ERASES            ;erase to bottom of screen
        LD      HL,MM1MSG         ;'(I)NDEX File, (M)AP or (T)RSDOS'
        CALL    VDLINE
MMU1    CALL    KBWAIT            ;wait for key to be pressed
        RES     5,A               ;convert l.c. to u.c.
        CP      'I'               ;'I' entered?
        JP      Z,FN01            ;start again and get file spec
        CP      'M'               ;'M' entered?
        JP      Z.ENDIN5          ;jp to TRSDOS and load MAP
        CP      'T'               ;'T' entered?
        JP      Z.JP2DOS          ;jp to TRSDOS
        JR      MMU1              ;loop until I,M or T entered
MM1MSG  DEFM    '(I)ndex File, (M)AP or (T)RSDOS'
        DEFB    3
;-----------------------------------------------------------------------
MHELP   LD      BC.6*256          ;mini-help screen
        CALL    POSCUR            ;r7:cl
        LD      HL,HMSG2          ;display help msg's
        CALL    VDLINE            ;msg's are at end of this routine
        LD      A,0DH
        CALL    VDCHAR            ;skip line
        LD      HL,HMSG4
        CALL    VDLINE
        LD      A,0DH
        CALL    VDCHAR            ;skip line
        LD      A,0DH
        CALL    VDCHAR            ;skip line
        LD      HL,HMSG6
        CALL    VDLINE
        LD      HL,HMSG8
        CALL    VDLINE
        LD      B,PENULN          ;next to bottom line
        LD      C,0
        CALL    POSCUR            ;r23:cl Mod II, r15:Cl Mod III
        CALL    TAPENT            ;Tap enter to continue
        JP      FN02              ;jp back and get file spec
HMSG2   DEFM    'Format:  SOURCE FILE,INDEX FILE;OPTIONS'
        DEFB    0DH
HMSG4   DEFM    'Only the source file specification is required'
        DEFB    0DH
HMSG6   DEFM    'Options: R = record format; P = paragraph format'
        DEFB    0DH
HMSG8   DEFM    '         M= return to MAP Master Menu; T = return to TRSDOS'
        DEFB    0DH
;-----------------------------------------------------------------------
                                  ;ENDIN
                                  ;--------------------------------
                                  ;end input file
ENDIN   CALL    WRITER            ;store last signature sector
```

Listing 8 continued

Listing 8 continued

```
                LD      DE,INDCB            ;DE->source file DCB
                CALL    CLOSE               ;close source file
                JP      NZ.ERROR            ;jp if error
                LD      DE,MAPDCB           ;DE->index file DCB
                CALL    CLOSE               ;close index file
                JP      NZ.ERROR            ;jp if error
                LD      A,(ENDCMD)          ;get end cmd
                CP      'T'                 ;return to TRSDOS?
                JP      Z.JP2DOS            ;jp to TRSDOS if true
        ENDIN5  LD      HL,MAPCMD           ;return to MAP Master Menu
                JP      DOSCMD              ;jp to TRSDOS and execute cmd
        MAPCMD  DEFM    'MAP'
                DEFB    0DH
        ;------------------------------------------------------------
        ;------------------------------------------------------------
                                            ;buffers used by M3
                                            ;-------------------------------
                                            ;general purpose variables
        ANSWER  DEFB    0                   ;one byte key in variable (Y/N etc)
                                            ;not used by MINDEX but must be
                                            ;present in public or there will
                                            ;be a link error
        CMDLFS  DEFB    0                   ;cmd line file spec (Y/N)
                                            ;indicates whether file spec
                                            ;was entered along with MINDEX cmd
        ENDCMD  DEFB    0                   ;end cmd (M=MAP, T=TRSDOS)
        FORMAT  DEFB    0                   ;indexing format (R=record, P=para)
        KEYBUF  DEFS    80                  ;used by MINDEX for file spec buffer
                DEFB    0DH                 ;precautionary EOR
        ;------------------------------------------------------------
                                            ;hash routine variables
        FBYTE   DEFW    0                   ;hash value of first byte of triplet
        MBYTE   DEFW    0                   ;hash value of middle byte of triplet
        LBYTE   DEFW    0                   ;hash value of last byte of triplet
        SCROSS  DEFB    0                   ;sector cross variable
                                            ;if 0 then sector boundary is not
                                            ;crossed by a record
                                            ;if >0 then the value in SCROSS
                                            ;is equal to the no of bytes in record
                                            ;prior to end of sector
        BICT    DEFB    1                   ;bit points to sector (0 to 7)
                                            ;
                                            ;disk I/O buffers
        RDERR   DEFB    0                   ;read error storage
        INBUF   DEFS    256                 ;source file buffer
                DEFB    0DH                 ;precautionary EOR for INBUF
        MAPBUF  DEFS    256                 ;index file buffer
                DEFB    0DH                 ;precautionary EOR for MAPBUF
        ;------------------------------------------------------------
                                            ;HEADER buffer
                                            ;-------------------------------
                                            ;byte  label   purpose
                                            ; 0- 9 HSEC    'MINDEX' and version
                                            ;10-10         0DH
                                            ;11-11 HSECPF  format: 0DH record
                                            ;              '.' paragraph
                                            ;12-19         blank (for future use)
                                            ;20-43 HFN     source file spec
                                            ;
        HSEC    DEFM    'MINDEX 1.2'
                DEFB    0DH
        HSECPF  DEFB    0DH
                DEFM    '        '
        HFN     DEFM    '                        '
                DEFB    0DH                 ;end of file spec
                DEFB    0DH                 ;backup end of file spec
        ;------------------------------------------------------------
                                            ;MINDEX (M3) messages
                                            ;-------------------------------
        MSGSFN  DEFM    'Source and Index File must have'
                DEFM    ' different names '
                DEFB    0DH
        MSGOP   DEFM    'FILES OPENED'
                DEFB    0DH
        DSPRD   DEFM    'READ COMPLETE'
                DEFB    0DH
        TITLE   DEFM    'MAP INDEX PROGRAM Version 1.2'
                DEFB    3
        CPYMSG  DEFM    '(c) Copyright 1983 Softshell'
                DEFB    3
        FNMSG   DEFM    'File Name:'
                DEFB    3
        FNMSG2  DEFM    'Enter name of file to be indexed, or ? for help'
                DEFB    0DH
        FNEMSG  DEFM    'Error in file name format'
                DEFB    0DH
        OOKMSG  DEFM    'Opened'
                DEFB    3
        MSG09   DEFM    'Source File:     '
                DEFB    3
        MSG10   DEFM    'MAP Index File:  '
                DEFB    3
        PARMSG  DEFM    'Paragraph Format'
                DEFB    3
        RECMSG  DEFM    'Record Format'
                DEFB    3
        OPEMSG  DEFM    'Error in Options Code'
                DEFB    0DH
        MSGSER  DEFM    'Error in opening Source File: '
                DEFB    03
        MSGMER  DEFM    'Error in opening MAP Index File: '
                DEFB    03
                DEFB    0DH
        IDXSEC  DEFM    'Indexing Sector: '
                DEFB    3
        ;------------------------------------------------------------
                END     BEGIN
```

Program Listing 9. Assembly-language listing of S3/SRC.

```
Ln #            Source Line

        ;S3/SRC   VERSION 1.2
        ;Jun 15, 1983
        ;------------------------------------------------------------
        ;SEARCH INFORMATION RETRIEVAL PROGRAM
        ;MAIN ROUTINE
        ;COPYRIGHT 1983, SOFTSHELL CORPORATION   ALL RIGHTS RESERVED
        ;------------------------------------------------------------
        SEARCH  PSECT                       ;begin relocatable program section
        ;------------------------------------------------------------
                                            ;public routines
                                            ;-------------------------------
                PUBLIC  MENU02B             ;mini-menu
                PUBLIC  OPNOUT              ;open output file
                PUBLIC  OUTREC              ;write record to output file
                PUBLIC  READIN              ;read input (source) file record
                                            ;
                                            ;public variables
                                            ;-------------------------------
                PUBLIC  ANSWER              ;1 byte switch
                PUBLIC  BACKSP              ;backspace through file routine
                PUBLIC  BEGIN               ;start of SEARCH program
                PUBLIC  CFILE               ;current file name
                PUBLIC  CMDLFS              ;cmd line file spec (Y/N)
                                            ;indicates whether file spec also entered
                                            ;along with SEARCH cmd
                PUBLIC  CODE
                PUBLIC  EOF                 ;end of file switch
                PUBLIC  INBUF               ;input (source) file record buffer
                PUBLIC  INBUFE              ;last byte in input record (256th)
                PUBLIC  INBUFU              ;byte after INBUFE
                PUBLIC  KEYBUF              ;keybuf used to enter MINDEX file spec
                PUBLIC  LSTSEC              ;
                PUBLIC  LINENR
                PUBLIC  MAPBUF
                PUBLIC  MAXMAP
                PUBLIC  NXTMAP
                PUBLIC  OFNAME              ;output file name
                PUBLIC  PARFOR              ;paragraph format routine
                PUBLIC  RECFOR              ;record format routine
                PUBLIC  MODE                ;display or list mode
                PUBLIC  NINSET              ;no. of spaces for inset
                PUBLIC  NXTREC              ;absolute position of next record
                PUBLIC  OFCOND              ;output file status (O, R or A)
                PUBLIC  OFNAME              ;output file name buffer
                PUBLIC  OFSTAT              ;output file status (Open or Closed)
                PUBLIC  OMODE               ;output mode (C or M)
                PUBLIC  PRMODE              ;print mode (C or M)
                PUBLIC  PSEOR               ;pseudo EOR char (if para format then = '.'
                                            ;                 if rec. format then = 0DH
                PUBLIC  RECBUF              ;record buffer (record found during search)
                PUBLIC  SDMODE              ;search-display mode
                PUBLIC  SINSET              ;storage for inset value
                PUBLIC  SPWORD              ;space or word
                                            ;
                                            ;public messages
                                            ;-------------------------------
                PUBLIC  MSGEOF              ;'End of File' msg
                PUBLIC  SEAMSG              ;'Searching' msg
        ;------------------------------------------------------------
                                            ;calls to MODII or MODIII
                                            ;-------------------------------
                EXTERN  BOTTOM              ;position cursor to last line of screen
                EXTERN  CLOSE               ;close file (DE->DCB)
                EXTERN  CURPOS              ;determine cursor pos (coor:BC)
                EXTERN  DISBRK              ;disable break (for Model II)
                EXTERN  DIVIDE              ;divide A into HL
                EXTERN  DOSCMD              ;jp to TRSDOS and execute command
                EXTERN  ERASEL              ;erase to end of line
                EXTERN  ERASES              ;erase to end of screen
                EXTERN  ERRDSP              ;MAP error display routine
                EXTERN  ERROR               ;TRSDOS error msg routine
                EXTERN  HOME                ;position cursor to top line of screen
                EXTERN  INIT                ;initialize file (create/open, pos to 1st rec)
                                            ;                (DE->DCB, HL->rec buf)
                EXTERN  JP2DOS              ;jp to TRSDOS
                EXTERN  KBCHAR              ;get char (or null) from keyboard
                EXTERN  KBINIT              ;initialize keyboard (Mod II)
                EXTERN  KBLIN3              ;get line from keyboard and end with 03
                EXTERN  KBLINE              ;get line from keyboard and end with 0DH
                EXTERN  KBWAIT              ;wait for char from keyboard
                EXTERN  KEYIN               ;get search key(s) form keyboard
                EXTERN  MOVELN              ;move line from screen to buffer
                                            ;HL->buf, D=no. char, BC=cursor coor
                EXTERN  NEWLN               ;role screen and pos cur to next to last line
                EXTERN  OPEN                ;open file (DE->DCB, HL->record buffer)
                EXTERN  OPINIT              ;open and initialize file
                                            ;          (DE->DCB, HL->record buffer)
                EXTERN  POSCUR              ;position cursor (coordinates in BC)
                EXTERN  POSEOF              ;position to end of file
                EXTERN  POSN                ;position to record in file
                EXTERN  PRINIT              ;initialize printer
                EXTERN  PRLINE              ;print line (HL->buf, B=no. char, C=end char)
                EXTERN  PRSCN               ;print entire screen
                EXTERN  PRTBLN              ;print last line of text
                EXTERN  PRTREC              ;print current record
                EXTERN  READDR              ;read file: direct access
                EXTERN  READNX              ;read next record in file
                EXTERN  SCRPRO              ;set scroll protect line (A=line no.)
                EXTERN  SETILR              ;set INLRR to true last record
                EXTERN  SETMLR              ;set MAPLLR to true last record
                EXTERN  STMEND              ;store top of memory in MEMEND
                EXTERN  VDCHAR              ;display char in A register
                EXTERN  VDCLS               ;clear screen
                EXTERN  VDINIT              ;initialize video I/O (for Model II)
                EXTERN  VDLINE              ;display line (HL->msg)
                EXTERN  WRITNX              ;write next record to file
        ;............................................................
                                            ;MODII/III buffers and I/O parameters
                                            ;...............................
                EXTERN  INDCB               ;INput (source) file DCB
                EXTERN  IDCBRL              ;Input file DCB Record Length
                EXTERN  IDCBCR              ;Input file DCB Current Record
                EXTERN  IDCBLR              ;Input file DCB Last Record
                EXTERN  INLLR               ;true last record of source file
                EXTERN  IDCBOS              ;Input file DCB xxxxxxx
                EXTERN  MAPDCB              ;MAP (index) file DCB
                EXTERN  MDCBRL              ;Map file DCB Record Length
                EXTERN  MDCBCR              ;Map file DCB Current Record
                EXTERN  MDCBLR              ;Map file DCB Last Record
                EXTERN  MAPLLR              ;true last record of index file
                EXTERN  MDCBOS              ;Map file DCB xxxxxxxx
                EXTERN  MEMEND              ;contents ->top of memory
                EXTERN  OUTCHR              ;record buffer (1 byte) for output file
                EXTERN  OUTDCB              ;output file DCB
                EXTERN  PRTBUF              ;printer buffer (80 char)
                EXTERN  SORLN               ;start of record line no.
                EXTERN  STMAP               ;start of MAP index file (always last byte
                                            ;of SEARCH program)
                                            ;
                                            ;I/O parameters:
                EXTERN  ARROWD              ;down arrow
                EXTERN  ARROWL              ;left arrow
                EXTERN  ARROWR              ;right arrow
                EXTERN  ARROWU              ;up arrow
                EXTERN  BRKKEY              ;break key value
                EXTERN  CPYPOS              ;copyright notice position (52 or 36)
                EXTERN  PENULN              ;next to last line on screen (23 or 15)
                EXTERN  SWIDTH              ;screen width (80 or 64)
        ;
        ;------------------------------------------------------------
                                            ;calls to GR (general routines)
                                            ;-------------------------------
                EXTERN  APPEND              ;append on string to another
                EXTERN  BXDEC               ;binary to decimal conversion
                EXTERN  DECXB               ;decimal to binary conversion
                EXTERN  DLINE               ;display dotted line (e.g. ...........)
                EXTERN  DSPNBR              ;display 5-digit decimal no. (right justified)
                EXTERN  DSPNBRL             ;display 5-digit decimal no. (left justified)
                EXTERN  EATBNK              ;remove leading blanks
                EXTERN  FSPEC               ;move cmd line file psec to KEYBUF
                EXTERN  GETMFN              ;get MAP file names
                EXTERN  LINE                ;display line (from char in A reg)
                EXTERN  PACKFS              ;pack (compress) file spec in fixed table
                EXTERN  PARSEF              ;parse one file spec
                EXTERN  REC0                ;zero-out 256-byte record
                EXTERN  RECBL               ;blank-out 256-byte record
                EXTERN  TAPENT              ;'Tap enter to continue'
        ;------------------------------------------------------------
                                            ;GR buffers
                                            ;-------------------------------
                EXTERN  SNAM,SEXT,SPSW,SDDR     ;source file table
                EXTERN  INAM,IEXT,IPSW,IDDR     ;index file table
                EXTERN  ONAM,OEXT,OPSW,ODDR
        ;
        ;------------------------------------------------------------
                                            ;calls to S3DSP display routines
                                            ;-------------------------------
                EXTERN  BYUPDN              ;bypass up-down switch
                EXTERN  LIST                ;list routine
                EXTERN  DSP                 ;display current record on CRT
        ;------------------------------------------------------------
                                            ;calls to S3OPT options routine
                                            ;-------------------------------
```

Listing 9 continued

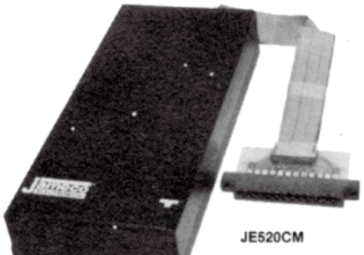
✓ See List of Advertisers on Page 291

Listing 9 continued

```
        EXTERN  OPT             ;options routine
;------------------------------------------------------------
                                ;calls to S3OPEN open files
                                ;-------------------------------
        EXTERN  GETFILE         ;get file spec, open files and
                                ;then load index into memory
;------------------------------------------------------------
;------------------------------------------------------------
                                ;SEARCH main segment
                                ;syntax:
                                ;SEARCH src file,ind file,out file;options
                                ;options:
                                ; T=return to TRSDOS or M=return to MAP
                                ; M is default
BEGIN   PUSH    HL              ;HL->cmd line buffer
                                ;save while top of memory is found and
                                ;break is disabled
        CALL    STMEND          ;set memory end (as many sig sectors as
                                ;possible will be stored in memory from
                                ;end of program to top of memory)
        CALL    DISBRK          ;disable break on Mod II
        POP     HL              ;HL->cmd line buffer
        CALL    FSPEC           ;get cmd line file spec if any
                                ;load file spec into KEYBUF
                                ;if file spec present LD CMDLFS with 'Y'
        LD      A,0
        CALL    SCRPRO          ;no subwindow
        CALL    VDCLS           ;erase screen
        LD      HL,TITLE        ;point to title
        CALL    VDLINE          ;display title
        LD      BC,CPYPOS       ;copyright notice coordinates (Mod II/III)
        CALL    POSCUR          ;rl:c52  Model II, rl:c  Model III
        LD      HL,CPYMSG       ;point to copyright notice
        CALL    VDLINE          ;display copyright notice
        LD      BC,1*256        ;rl:cl
        CALL    POSCUR
        LD      A,'-'           ;select '-' char for LINE routine
        CALL    LINE            ;draw dashed line under title
;
                                ;initialize varaibles:
        LD      A,'N'
        LD      (BYUPDN),A      ;do not bypass pause after displaying
                                ;a record
        LD      A,'R'           ;initial search dsp mode = record
        LD      (SDMODE),A      ;initialize search dsp mode
        LD      A,'M'           ;initial print mode = manual
        LD      (PRMODE),A
        LD      (OMODE),A       ;initial output mode = manual
;------------------------------------------------------------
                                ;GETFSP
                                ;-------------------------------
GETFSP  CALL    GETFILE         ;get file spec, open files and
                                ;then load index into memory
;------------------------------------------------------------
                                ;MASTLP
                                ;-------------------------------
                                ;master loop
                                ;after each search the program returns
                                ;to this point
MASTLP  LD      A,0
        LD      (EOF),A         ;set EOF=0 (i.e. not end of file)
        CALL    HOME            ;rl:cl
        CALL    ERASEL          ;erase line
        LD      HL,MSGKEY       ;'Enter Key'
        CALL    VDLINE
        LD      BC,30           ;rl:c31
        CALL    POSCUR
        LD      HL,CFILE        ;current source file name
        CALL    VDLINE          ;display current file
        LD      A,3
        CALL    SCRPRO          ;set scroll protect at value in A
        LD      BC,2*256        ;r3:cl
        CALL    POSCUR
        LD      A,'-'           ;select '-' char for LINE routine
        CALL    LINE            ;draw dashed line
        CALL    GETKEY          ;get key(s)
        CP      BRKKEY          ;was break entered?
        JP      NZ,CALSYN       ;jp to CLASYN if break not entered
        CALL    MENU02          ;call mini-menu
        CP      'C'             ;change file spec?
        JP      Z,GETFSP        ;loop back and get file spec
        CP      'O'             ;options?
        JP      Z,MASTLP        ;loop back and get search key
        CP      'M'             ;MAP?
        JP      Z,DOSCMD        ;jp to MAP
        CP      'T'             ;TRSDOS?
        JP      Z,JP2DOS        ;jp to TRSDOS
        JP      MASTLP          ;loop back and get search key
CALSYN  CALL    SYNTAX          ;check syntax of search key
        CP      0               ;0=improper syntax
        JP      Z,MASTLP        ;try again if improper
        CALL    HSHKEY          ;hash search key
;------------------------------------------------------------
FIND2   CALL    SEAMSG          ;display 'Searching' msg
        LD      HL,1            ;start at 1 since header = 0
        LD      (CMSEC),HL      ;current map sector = 1
        LD      HL,STMAP        ;HL-> start of index in memory
        LD      (NXTMAP),HL     ;(NXTMAP) -> 1st map sec
        LD      IX,0000H        ;IX -> 1st input sec
        CALL    NEWLN
;
FIND4   NOP                     ;get map sector and bits
                                ;next op stored in NXTOP
        CALL    KBCHAR          ;checks for break
        CP      BRKKEY          ;<break>?
        JP      Z,MASTLP        ;jp to start of loop if break
        LD      HL,HSHBUF       ;HL -> HSHBUF (search key signature buf)
        LD      A,255
        LD      (TBYTE),A       ;set TBYTE = 255
        LD      A,0
        LD      (TTBYTE),A
        LD      A,253           ;and
        LD      (NXTOP),A       ;first op = and
        LD      C,0             ;init length = 0
;...
FIND4F  LD      IY,(NXTMAP)     ;IY-> byte 0 map sector
        LD      A,(HL)          ;A(i) = hashcode(i)
        CP      253             ;and op?
        JR      Z,SNXTOP
        CP      254             ;or op?
        JR      Z,SNXTOP
        JP      FIND4H
SNXTOP  NOP                     ;save next op
        LD      D,A             ;D has future oper
        LD      A,(NXTOP)       ;get current oper
        LD      E,A             ;E has current oper
        LD      A,D
        LD      (NXTOP),A       ;store NXTOP
        LD      A,D             ;check future oper
        CP      253             ;and?
        JP      Z,FDAND
        CP      254             ;or?
        JR      FDOR
FDAND   NOP                     ;and op
        JP      FIND4I
FDOR    NOP                     ;or op
        LD      A,(TTBYTE)
        LD      B,A             ;B has TTBYTE
        LD      A,(TBYTE)
        OR      B               ;or TBYTE with TTBYTE
        LD      (TTBYTE),A      ;save new TTBYTE
        LD      A,255
        LD      (TBYTE),A       ;reinit (TBYTE)
        JP      FIND4I
FIND4H  LD      D,0             ;HSHBUF byte not op
        LD      E,A             ;so see if bit pattern
        ADD     IY,DE           ;present in current sec
        LD      A,(TBYTE)
        AND     (IY+0)          ;and TBYTE with HSH byte
        LD      (TBYTE),A       ;save new value
FIND4I  INC     C               ;inc length counter
        LD      A,(WDLEN)
        CP      C               ;has (WDLEN) been reached
        JP      Z,FIND4K
        INC     HL
        JP      FIND4F          ;get next HSHBUF byte
;...
FIND4K  NOP                     ;end of map sector search
        LD      A,(TTBYTE)
        LD      D,A
        LD      A,(TBYTE)
        OR      D
        JP      FIND4T
FIND4P  LD      A,D             ;last op = or
        OR      E
FIND4T  CP      0               ;any key found?
        JP      Z,NOKEY
        LD      (TBYTE),A       ;key found, save (TBYTE)
        JP      FIND6
NOKEY   INC     IX              ;no key in MAP sector
        INC     IX              ;so bump source file pointer 8 sectors
        INC     IX
        INC     IX
        INC     IX
        INC     IX
        INC     IX
        INC     IX
        JP      FIND8
;...
;------------------------------------------------------------
;............................................................
                                ;FIND6
                                ;...............................
                                ;signature screen positive
                                ;so search appropriate src file sectors
FIND6   NOP
;       CALL    KBBRK           ;ADD LATER !!!
;       JP      NZ,MASTLP
        LD      A,0
        LD      (BICT),A        ;BICT = current bit being scanned
FIND6D  LD      A,(TBYTE)
        RRCA                    ;move bit into carry
        LD      (TBYTE),A
        JP      NC,FIND6L       ;if bit = 0, don't search sector
        CALL    GETSEC          ;get sector and search it for key(s)
        CP      BRKKEY          ;<break>?
        JP      Z,MASTLP        ;abort search if <break> pressed
FIND6L  INC     IX              ;bump input rec #
        LD      A,(BICT)
        CP      7               ;have all 8 sectors been checked?
        JP      Z,FIND8         ;jp if all 8 sectors checked
        INC     A
        LD      (BICT),A        ;increment bit count
        JR      FIND6D          ;loop until all 8 sectors checked
;............................................................
                                ;FIND8
                                ;...............................
                                ;increment pointers
FIND8   LD      HL,(NXTMAP)     ;HL ->next map sector in memory
        LD      DE,256
        ADD     HL,DE           ;next map sector in memory bumped by 1 sector
        LD      (NXTMAP),HL     ;store result
        LD      DE,(MAXMAP)     ;check if NXTMAP > MAXMAP
        XOR     A
        SBC     HL,DE
        JP      P,FIND10
        JR      Z,FIND10
                                ;next map sector still in memory
        LD      HL,(CMSEC)      ;HL -> current MAP sector
        INC     HL              ;inc CMSEC
        LD      (CMSEC),HL
        JP      FIND4           ;loop and search next 8 src file sectors
;............................................................
                                ;FIND10
                                ;...............................
                                ;required MAP sector not in memory
FIND10  LD      HL,(CMSEC)      ;HL -> current MAP sector on disk
        LD      DE,(MAPLLR)     ;DE = last sector in MAP file
        XOR     A
        SBC     HL,DE
        JP      P,FIND12        ;end of search if CMSEC > MAPLLR
        JR      Z,FIND12        ;end of search if CMSEC = MAPLLR
        LD      HL,(CMSEC)
        INC     HL              ;bump current sec no.
        LD      (CMSEC),HL
        CALL    RDMSEC          ;read MAP sector pointed to by CMSEC
                                ;transfer from MAPBUF to MAXMAP position
        LD      HL,MAPBUF
        LD      DE,(MAXMAP)
        LD      BC,256
        LDIR                    ;move next sec to top of MAP
        LD      HL,(MAXMAP)
        LD      (NXTMAP),HL     ;NXTMAP -> next MAP sector in memory
        JP      FIND4           ;loop and search next 8 sectors
;
FIND12  JP      MASTLP          ;end of search
;------------------------------------------------------------
                                ;GETKEY
                                ;-------------------------------
                                ;input search key(s)
                                ;entry: no parameters
                                ;exit: A = <break> or EOR=0DH
                                ;reg altered: AF,HL,DE,BC
GETKEY  LD      BC,1*256+0
        CALL    POSCUR          ;r2;cl
        CALL    ERASEL          ;erase to end of line
        LD      B,60            ;max key length = 60
        LD      HL,KEYBUF       ;HL->search key buffer
        CALL    KEYIN           ;keyin routine for search key
        CP      BRKKEY          ;was break key pressed?
        JP      NZ,GKEY1
                                ;break was pressed
        LD      BC,3*256
        CALL    POSCUR          ;r4:cl
        CALL    ERASES          ;erase to end of screen
        LD      A,BRKKEY        ;A returns with break key value
        RET
;
GKEY1   LD      HL,KEYBUF       ;convert upper case letters to lower case
GKEY2   LD      A,(HL)          ;get char from search key
        CP      0DH             ;EOR?
        JR      Z,GKEY8         ;exit loop if end of search key
        CP      91              ;'Z'
        JP      P,GKEY4
        JR      Z,GKEY4         ;needed??
        CP      65              ;'A'
        JP      M,GKEY4         ;bypass if < 'A'
        OR      32              ;convert to lower case
        LD      (HL),A          ;overwrite upper case with lower case
GKEY4   INC     HL
        JR      GKEY2           ;loop until EOR
GKEY8   LD      HL,0
        LD      (LLRN),HL       ;init (LLRN)=0
        LD      BC,3*256
        CALL    POSCUR          ;r4:cl
        CALL    ERASES          ;erase to end of screen
        LD      A,0DH           ;return with A = EOR
        RET
;------------------------------------------------------------
                                ;SYNTAX
                                ;-------------------------------
                                ;check search key(s) for proper syntax
                                ;KEYIN routine stores
                                ;         + operator as 253
                                ;         | operator as 254
                                ;each key must be 3 or more char long
                                ;entry: no parameters passed
                                ;exit: A = validity of key(s)
                                ;      A = 0, bad key; A = 1 proper key(s)
                                ;reg altered: AF, BC, DE, HL
SYNTAX  LD      HL,KEYBUF       ;HL-> search key buffer
SYNRST  LD      C,0             ;C = key length (init to 0)
```

Listing 9 continued

VOICE SYNTHESIZER FOR APPLE AND COMMODORE



JE520CM

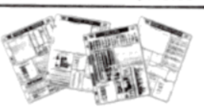
• Over 250 word vocabulary-affixes allow the formation of more than 500 words • Built-in amplifier, speaker, volume control, and audio jack • Recreates a clear, natural male voice • Plug-in user ready with documentation and sample software • Case size: 7¼"L × 3¼"W × 1-3/8"H

APPLICATIONS:
- Security Warning
- Teaching
- Instrumentation
- Telecommunication
- Handicap Aid
- Games

The JE520 VOICE SYNTHESIZER will plug right into your computer and allow you to enhance almost any application. Utilizing National Semiconductor's DIGITALKER[tm] Speech Processor IC (with four custom memory chips), the JE520 compresses natural speech into digital memory, including the original inflections and emphases. The result is an extremely clear, natural vocalization.

Part No.	Description	Price
JE520CM	For Commodore 64 & VIC-20	$114.95
JE520AP	For Apple II, II+, and *II*e	$149.95

Micro-Logic Corp. MICRO-CHARTS



• Fully decoded data • Instant access • 2-sided, totally comprehensive • Compact 8½ x 11 in. durable credit card plastic • Perfect for programmers & engineers • Clear & concise tables for: full instruction set, disassembly, ASCII, base conversion, effect of flags, compare vs. jump, interrupt structure, pinout, cycle times, diagrams, bug notes, & much more...

PART NO.	REFERENCE	PRICE
ML-Z80	Z80 CPU	$5.95
ML-8080A	8080A/8085A	$5.95
ML-6502	6502 (65XX)	$5.95
ML-8048	8048, Relatives	$5.95
ML-7400	5400/7400 TTL Pin-Outs	$5.95

BOOKS

30001	National CMOS Data Book (1981) (640 pages) 74C, CD4000, and A/D Converters	$6.95
30003	National Linear Data Book (1982) (1376 pages) LF, ADC, DAC, LH Series	$11.95
30008	National Memory Data Book (1980) (464 pages) RAMs, ROMs, PROMs, EPROMs Series	$6.95
30009	Intersil Data Book (1983) (1356 pages) Complete line.	$9.95
30010	National Audio/Radio Handbook (1980) (240 pages) Pre-Amps, AM, FM & FM Stereo, Power Amps	$5.95
30012	National PAL Data Book (1982) (176 pages) Application Notes, Linear Briefs, etc.	$5.95
30013	Zilog Data Book (1983) (641 pages) Microprocessors and Support Chips	$7.95
210830	Intel Memory Components Handbook (1983) (798 pages) Contains all Application Notes, Article Reprints, Data Sheets, and other design information on Intel's RAMs, EPROMs, E²PROMs & Bubble Memories.	$14.95
210844	Intel Microprocessor & Peripheral Handbook (1983) (1027 pages) Contains Data Sheets on all of Intel's Microprocessors and Peripherals.	$14.95

4-Digit Fluorescent Alarm Clock Kit

• Bright 4-digit 0.5" high display • 10 minute snooze alarm
• AM/PM indicator

The JE750 Clock Kit is a versatile 12-hour digital clock with 24-hour alarm. The clock has a bright 0.5" high blue-green flourescent display. The 24-hour alarm allows the user to disable the alarm and immediately re-enable the alarm to activate 24 hours later. The kits includes all documentation, components, case and wall transformer. Size: 6-5/8"L × 3¼"H × 1¾"D.

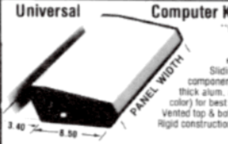
JE750 Alarm Clock Kit **$29.95**

Jameco Electronics Digital Thermometer Kit

Dual sensors — switch controls for indoor/outdoor or dual monitoring — can be extended to 500 feet. Continuous LED .8" ht. display. Range: -40°F to 199°F, -40°C to 100°C. Accuracy ±1° nominal. Calibrate for Fahrenheit/Celsius. Simulated walnut case. AC wall adapter included. Size: 6⅜"L x 3⅞"H x 1⅞"D.

JE300 ...... **$39.95**

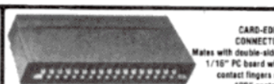
Universal Computer Keyboard Enclosures



"DTE" Blank Desk-Top Enclosures are designed for easy modification. High strength epoxy molded end pieces in mocha brown finish. Sliding rear/bottom panel for service/component access. Top/bott. panels .080" thick alum. alodine type 1200 finish (gold tint color) for best paint adhesion after modification. Vented top & bottom panels for cooling efficiency. Rigid construction provides unlimited applications. Assembly instructions included.

DTE-8	Panel Width 7.5"	$24.95
DTE-11	Panel Width 10.13"	$27.95
DTE-14	Panel Width 13.5"	$29.95
DTE-20	Panel Width 19.25"	$34.95
DTE-22	Panel Width 21.375"	$39.95

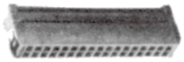
CARD-EDGE CONNECTORS



CARD-EDGE CONNECTOR Mates with double-sided 1/16" PC board with contact fingers on .100" centers

Part No.	No. of Contacts/ Conductors	1-9	10-99	100+
C20	20	2.39	2.09	1.89
C26	26	2.69	2.39	2.09
C34	34	3.29	2.95	2.59
C40	40	3.69	3.29	2.89
C50	50	4.39	3.95	3.39

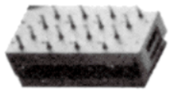
SOCKET CONNECTORS



SOCKET CONNECTOR Mates with 2 rows of .025" sq. or dia. posts on patterns of .100" centers and shielded receptacles

Part No.	No. of Contacts/ Conductors	1-9	10-99	100+
S20	20	1.39	1.25	1.09
S26	26	1.69	1.49	1.35
S34	34	2.05	1.79	1.59
S40	40	2.29	2.09	1.79
S50	50	2.69	2.39	2.09
S60	60	3.39	2.95	2.59

SOLDER TRANSITION CONNECTORS



FOR PERMANENT TERMINATION TO PRINTED CIRCUIT BOARDS

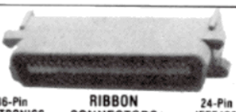
Part No.	No. of Contacts	1-9	10-99	100+
ST20	20	1.15	.99	.89
ST26	26	1.25	1.15	.99
ST34	34	1.49	1.35	1.15
ST40	40	1.69	1.55	1.35
ST50	50	1.89	1.69	1.49

MALE CONNECTOR

Mates with Socket Connector .10" × .10"

Part No.	No. of Contacts/ Conductors	1-9	10-99	100+
M20	20	4.09	3.65	3.19
M26	26	4.49	3.95	3.49
M34	34	4.95	4.49	3.95
M40	40	5.49	4.89	4.29
M50	50	6.49	5.75	4.95

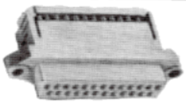
RIBBON CONNECTORS



36-Pin CENTRONICS — 24-Pin IEEE488

Part No.	No. of Contacts/ Style	1-9	10-99	100+
CEN14M	14 Male	6.39	5.49	4.49
CEN14F	14 Female	6.19	5.29	4.29
CEN24M	24 Male	7.95	6.95	5.95
CEN24F	24 Female	7.69	6.79	5.75
CEN36M	36 Male	8.69	7.59	6.49
CEN36F	36 Female	9.75	8.95	7.95

D-SUB CONNECTORS



Part No.	No. of Contacts/ Style	1-9	10-99	100+
CDE9P	9 Male	2.95	2.65	2.29
CDE9S	9 Female	3.39	3.09	2.69
CDA15P	15 Male	3.95	3.55	3.09
CDA15S	15 Female	4.75	4.19	3.69
CDB25P	25 Male	5.49	4.79	4.25
CDB25S	25 Female	5.95	5.49	4.85
CDC37P	37 Male	7.79	6.95	6.09
CDC37S	37 Female	8.95	7.95	6.95

28AWG FLAT GREY CABLE

Part No.	No. of Contacts/ Conductors	Connector & Wire Type	Price Per Foot
171-9	9	28AWG Flat Stranded	.19
171-14	14	28AWG Flat Stranded	.25
171-15	15	28AWG Flat Stranded	.27
171-16	16	28AWG Flat Stranded	.29
171-20	20	28AWG Flat Stranded	.36
171-24	24	28AWG Flat Stranded	.45
171-25	25	28AWG Flat Stranded	.47
171-26	26	28AWG Flat Stranded	.49
171-34	34	28AWG Flat Stranded	.62
171-36	36	28AWG Flat Stranded	.65
171-37	37	28AWG Flat Stranded	.69
171-40	40	28AWG Flat Stranded	.75
171-50	50	28AWG Flat Stranded	.89

CUSTOM ASSEMBLIES

Use the part numbers from the connectors and cable to order your own custom assembled cables.

EXAMPLE: If you desire a 25-foot cable with a male "Centronics" connector on one end, and a female "Centronics" connector on the other end, you would order:

CEN36M-25'-CEN36F CUSTOM

$8.69 (CEN36M) + 9.75 (CEN36F) =	$18.44
.65 × 25 (25 feet 36 cond. cable) =	$16.25
2.00 Set-up charge on all custom cables =	$ 2.00
This (CEN36M-25'-CEN36F CUSTOM) Cable would cost	$36.69

Please specify "CUSTOM" after the part no. to ensure your order will be filled correctly. (**Important**: Please specify cable in FEET, not inches)

ALL CUSTOM ASSEMBLIES MUST BE PREPAID BEFORE ASSEMBLY. THERE ARE NO RETURNS ON CUSTOM CABLE ASSEMBLIES. $7.00 SET-UP CHARGE (PER STYLE) ON ALL CUSTOM CABLES

ACCESSORIES FOR APPLE* COMPUTERS

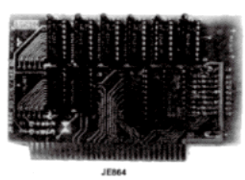
Numeric/Auxiliary Keypad for APPLE *II*e*

JE614

The JE614 is a newly introduced numeric/auxiliary keypad for the APPLE IIe.* It offers the flexibility of a 10-key pad and the convenience of 23 directly accessible functions. Screen manipulating functions make word processing a snap and cursor controls make the keypad ideal for VisiCalc** users. The JE614 Keypad is housed in a durable metal enclosure and is color-coordinated with your APPLE IIe computer. Operation of the keypad can begin within minutes from unpacking. Special functions include: Home, Clear, Clear to End of Screen, Scroll-Up, Scroll-Down, Tab, Delete, Left, Right, Up and Down. Each key has auto-repeat.

JE614 Assembled and Tested **$89.95**

Extended 80 Column/64K RAM Card for APPLE *II*e*



JE864

Now you can double the memory capacity and get an 80 column display format for your APPLE IIe computer at an affordable price. Just plug the JE864 card into your APPLE and expand your display to 80-characters per line. Perfect for word processing. The JE864 also features 64K bytes of additional memory to allow programming not possible with standard APPLE IIe computers.

Board: High density board design squeezes 64K bytes of RAM onto a 2½" × 4½" board • Fully tested to assure proper operation.

Uses: Word processing — displays 1000 more characters per screen • Extra memory allows running of extremely large programs • Ultra High Resolution Graphics (when software available)

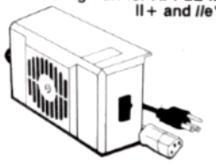
JE864 Assembled and Tested **$149.95**

Switching Power Supply for APPLE II, II+ and *II*e*

• Can drive four floppy disk drives and up to eight expansion cards • Short-circuit and overload protection • Fits inside Apple computer • Fully regulated: +5V @ 5A, +12V @ 3A, -5V @ .5A, -12V @ .5A • Apple-type plug in power cord included • Size: 9⅞"L × 3¾"W × 2¾"H • Weight: 2 lbs.

Part No. KHP4007 .... $79.95

Cooling Fan for APPLE II, II+ and *II*e*



• Snaps on the side of APPLE II, II+ and IIe* enclosures • Eliminates overheating problems, thereby boosting reliability and operation life of computer • Switch on front of fan serves as power switch for fan, computer, and provided extra outlet.

Part No. APF-1 .... $49.95

*APPLE and APPLE IIe are registered trademarks of APPLE Computers — **VisiCalc is a registered trademark of Visi Corp. Inc.

POWER SUPPLY +5VDC @ 7.5 AMP, 12VDC @ 1.5 AMP SWITCHING

Input: 115VAC, 50-60Hz @ 3 amp/230VAC, 50Hz @ 1.6 amp. Fan volt./power supply select switches (115/230VAC). Output: 5VDC @ 7.6 amp, 12VDC @ 1.6 amp. 8 ft. blk. pow. cord. 11½"W x 13¼"D x 3¾"H. Wt. 6 lbs.

Part No. PS94V0S .... $39.95 each

POWER SUPPLY 4-Channel Switching - Apple Compatible

Microprocessor, mini-computer, terminal, medical equipment and process control applications. Input: 90-130VAC 47-440Hz. Output: +5VDC @ 5A, -5VDC @ 1A; +12VDC @ 1A, -12VDC @ 1A. Line reg.: ±0.2%. Ripple: 30mV p-p. Load reg.: ±1%. Overcurrent protection. Adj: 5V main output ± 10%. 6-3/8"L x 1-7/8"W x 4-15/16"H. Wt. 1½ lbs.

Part No. FCS-604A .... $69.95 each

$10.00 Minimum Order — U.S. Funds Only
California Residents Add 6½% Sales Tax
Shipping — Add 5% plus $1.50 Insurance
Send S.A.S.E. for Monthly Sales Flyer!

Spec Sheets — 30¢ each
Send $1.00 Postage for your
FREE 1984 JAMECO CATALOG
Prices Subject to Change

MasterCard — Mail Order Electronics • Worldwide — **Jameco ELECTRONICS** — VISA®

1355 SHOREWAY ROAD, BELMONT, CA 94002

✓534

12/83 *PHONE ORDERS WELCOME — (415) 592-8097 Telex: 176043*

JE664 EPROM PROGRAMMER

8K TO 64K EPROMS — 24 AND 28 PIN PACKAGES

• Programs, validates, and checks for properly erased EPROMs • Emulates PROMs or EPROMs • RS232C Computer Interface for editing/program loading • Loads data into RAM by keyboard • Changes data in RAM by keyboard • Loads RAM from an EPROM • Compares EPROMs for content differences • Copies EPROMs • Power input: 115VAC, 60Hz, ◄ 10W power consumption • Enclosure: Color-coordinated, light tan panels w/molded mocha brown end pieces • Size: 15-5/8"L x 8¾"D x 3½"H • Wt.: 5¾ lbs.

JE664-A EPROM Programmer **$995.00**
Assembled & Tested (Includes JM16A Module)

JE665 — RS232C INTERFACE OPTION — The JE665 RS232C Interface Option implements computer access to the JE664's RAM. Sample software written in BASIC provided for TRS-80® Model I, Level II Computer. Baud rate: 9600. Word lgth: 8 bits - odd parity. Stop bits: 2. Option may be adapted to other computers. The JE665 can be interfaced to any computer with an RS232 port. Information is also provided for interfacing to any CP/M system with an RS232 port.

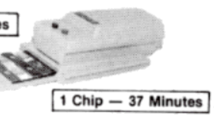
JE664-ARS EPROM Programmer w/JE665 Option **$1195.00**
Assembled and Tested (Includes JM16A Module)

EPROM JUMPER MODULES - The JE664's JUMPER MODULE (Personality Module) is a plug-in Module that pre-sets JE664 for proper programming pulses to the EPROM & configures EPROM socket connections for that particular EPROM.

P/N	EPROM	EPROM MANUFACTURER	PRICE
JM08A	2708	AMD, Motorola, National, Intel, TI (25V)	$14.95
JM16A	2716, TMS2516	Intel, Motorola, National, NEC, TI, (25V)	$14.95
JM16B	TMS2716	Motorola, TI (+5, -5, +12)	$14.95
JM32A	TMS2532	Motorola, TI (25V)	$14.95
JM32B	2732	AMD, Fujitsu, NEC, Hitachi, Intel (25V)	$14.95
JM32C	2732A (21V)	Fujitsu, Intel (21V)	$14.95
JM64A	MCM68764, MCM68L764	Motorola (21V)	$14.95
JM64B	2764	Intel (21V)	$14.95
JM64C	TMS2564	TI (25V)	$14.95
JM64D	HN482764G-4	Hitachi (21V)	$14.95

UV-EPROM Eraser

8 Chips — 51 Minutes



1 Chip — 37 Minutes

Erases 2708, 2716, 2732, 2764, 2516, 2532, 2564. Erases up to 8 chips within 51 minutes (1 chip in 37 minutes). Maintains constant exposure distance of one inch. Special conductive foam liner eliminates static build-up. Built-in safety lock to prevent UV exposure. Compact — only 9.00" x 3.70" x 2.60". Complete with holding tray for 8 chips.

DE-4 UV-EPROM Eraser ... **$79.95**
UVS-11EL Replacement Bulb **$16.95**

5¼" APPLE™ COMPATIBLE DISK DRIVE

• Uses Shugart SA390 mechanics • 143K formatted storage • 35 tracks — compatible with Apple controller • Complete with connector and cable — just plug into your disk controller card • Size: 6"L × 3⅞"W × 8-9/16"D • Weight: 4¼ lbs.

Part No. ADD-514 $195.95

8" FLOPPY DISK DRIVE

- Shugart 801R compatible
- Single-Sided
- 77 Tracks
- 400/800K Bytes Capacity
- Industry Standard

The **FDD100-8 8" Floppy Disk Drive** (Industry Standard) features single or double density. Recording mode: FM single, MFM double density. Transfer rate: 250K bits/sec. single density; 500K bits/sec. double density. The FDD100-8 is designed to work with the single-sided soft sectored IBM Diskette I, or eq. disk cartridge. Power: 115VAC @ 50-60Hz, +24VDC @ 1.7 amps max., +5VDC @ 1.2 amps max. Unit as pictured above (does not include case, power supply, or cables). Size: 8.55"W x 14"L x 4.5"H. Weighs 12 lbs. Incl. 96-pg. manual.

Part No. — Price
FDD100-8 .. $169.95 ea.

IBM MEMORY EXPANSION KIT

SAVE HUNDREDS OF $$$ BY UPGRADING MEMORY BOARDS YOURSELF!

Most of the popular memory boards allow you to add an additional 64K, 128K, 192K, or 256K. The IBM64K Kit will populate these boards in 64K byte increments. The kit is simple to install — just insert the nine 64K RAM chips in the provided sockets and set the two groups of switches. Directions are included.

IBM64K (Nine 200ns 64K RAMs) **$49.95**

EXPAND YOUR MEMORY

TRS-80 to 16K, 32K, or 48K

**Model 1 = From 4K to 16K Requires (1) One Kit
Model 3 = From 4K to 48K Requires (3) Three Kits
Color = From 4K to 16K Requires (1) One Kit

**Model 1 equipped with Expansion Board up to 48K Two Kits Required — One Kit Required for each 16K of Expansion —

TRS-16K3 *200ns for Color & Model III $12.95
TRS-16K4 *250ns for Model I $10.95

TRS-80 Color 32K or 64K Conversion Kit

Easy to install kit comes complete with 8 ea. 4164-2 (200ns) 64K dynamic RAMs & conversion documentation. Converts TRS-80 color computers with E circuit boards, & all new color computers to 32K. Minor modifications of 32K memory will allow the use of all the 64K of the dynamic RAM providing you have a FLEX DOS operating system.

TRS-64K2 .... **$44.95**

✓ See List of Advertisers on Page 291

Listing 9 continued

```
SYNLP   LD      A,(HL)          ;get char from search key buffer
        INC     HL              ;pt to next char in search key buffer
        CP      0DH             ;EOR? (end of search key?)
        JP      Z,SYNEND        ;exit loop if end of key
        CP      253             ;and ?
        JR      Z,SYNOP
        CP      254             ;or ?
        JR      Z,SYNOP
        INC     C               ;valid char so bump key length count
        JR      SYNLP           ;loop until end of key reached
SYNOP   LD      A,3             ;key terminated by operand
        CP      C               ;was key >= 3?
        JR      Z,SYNRST        ;key len = 3, so get next key
        JP      M,SYNRST        ;key len > 3 so get next key
SYNER   LD      BC,3*256        ;syntax error (key < 3 char)
        CALL    POSCUR          ;r4:cl
        CALL    ERASES          ;erase to end of screen
        LD      HL,THREE        ;'Each key must be three or more char'03
        CALL    VDLINE
        CALL    TAPENT          ;Tap enter to continue
        LD      A,0
        RET                     ;A=0 => bad syntax
SYNEND  LD      A,3             ;end of key(s) reached
        CP      C               ;check that last key >= 3
        JR      Z,SYNRET        ;len =3
        JP      M,SYNRET        ;len >3
        JR      SYNER           ;key <3 so jp to error msg
SYNRET  LD      A,1
        RET                     ;A=1 => proper syntax
;------------------------------------------------------------
                                ;HASHKEY
                                ;------------------------------
                                ;hash search key
                                ;search key in KEYBUF
                                ;triplet hash codes stored in HSHBUF
                                ;see M3 (index program) for hash algo.
                                ;entry: no parameters passed
                                ;exit: len of hsh code stored in WDLEN
                                ;reg altered: AF, BC, DE, HL, IX, IY
HSHKEY  LD      C,0             ;C = key len
        LD      IY,HSHBUF       ;IY -> search key signature buffer
        LD      IX,KEYBUF       ;IX -> search key buffer
HSH3    LD      L,(IX+0)        ;get char from search key
        LD      H,0             ;HL contains char
        LD      A,L
        CP      0DH             ;end of string?
        JP      Z,HSH8          ;if EOF then end hashing
        CP      ' '             ;blank?
        JR      NZ,HSH4         ;continue at HSH if not blank
        INC     IX              ;pt to next char
        JR      HSH3            ;loop until non-blank char reached
                                ;now get first 2 char of key.
HSH4    CALL    MUHL10          ;mult value in HL by 10
        LD      (MBYTE),HL      ;store in MBYTE
        INC     IX              ;pt to next char in search key
        LD      L,(IX+0)
        LD      H,0
        LD      (LBYTE),HL      ;store value of char in LBYTE
        INC     IX              ;pt to next char in search key
;............................................................
                                ;HSHLP
                                ;..............................
                                ;get triplet and hash it
HSHLP   LD      .,(MBYTE)       ;get 10ac2
        CALL    HL10            ;mult by 10
        LD      BYTE),HL        ;store 100ac1
;
        LD      HL,(LBYTE)      ;get ac3
        CALL    MUHL10          ;mult by 10
        LD      (MBYTE),HL      ;store 10ac2
;
        LD      A,(IX+0)        ;get char (cl) from search key
        INC     IX              ;inc IX to next char
        CP      0DH             ;EOR?
        JP      Z,HSH8
        CP      ' '
        JP      Z,HSH3          ;skip 3 hashes
        CP      253             ;and op?
        JR      Z,HSH5
        CP      254             ;or op?
        JR      Z,HSH5
        JR      HSH6
HSH5    LD      (IY+0),A        ;store op in hash buffer
        INC     IY              ;pt to next byte in hash buffer
        INC     C               ;inc hash buf len counter
        JP      HSH3            ;loop until end of search key
;............................................................
                                ;HSH6
                                ;..............................
                                ;hash triplet
HSH6    LD      L,A
        LD      H,0
        LD      (LBYTE),HL
        LD      DE,(MBYTE)
        ADD     HL,DE           ;10ac2+ac3
        LD      DE,(FBYTE)
        ADD     HL,DE           ;100ac1 + (10ac2+ac3)
        LD      A,251
        CALL    DIVIDE          ;A has modulo 251 result
        LD      (IY+0),A        ;store hash code
        INC     C               ;inc hash code len
        INC     IY
        JP      HSHLP           ;loop until end of search key(s)
;............................................................
                                ;HSH8
                                ;..............................
HSH8    LD      A,C             ;hashing complete
        LD      (WDLEN),A       ;save hash len
        RET
;------------------------------------------------------------
MUA10   NOP                     ;mult # in A by 10
                                ;BC and DE used
                                ;value returned in HL
        LD      H,0
        LD      L,A
MUHL10  NOP                     ;multiply # in HL by 10
        ADD     HL,HL           ;HL contains 2*no.
        LD      E,L
        LD      D,H
        ADD     HL,HL           ;4
        ADD     HL,HL           ;8
        ADD     HL,DE           ;10
        RET
;------------------------------------------------------------
                                ;GETSEC
                                ;------------------------------
                                ;get sector from source file
                                ;entry: no para
                                ;exit: sector from disk in INBUF
                                ;reg altered: AF,BC,DE,HL
GETSEC  LD      HL,INBUF        ;pt to source file rec buf
        DEC     HL              ;HL->last char of record before INBUF
        LD      A,(HL)
        LD      (B256),A        ;last byte of SINBUF
                                ;save this byte for analysis of records
                                ;that span sector
        PUSH    IX
        POP     BC              ;BC -> input rec to read
        LD      (CLRN),BC       ;store current LRN
        CALL    READ2           ;read two sectors from src file
        CALL    SECTOR
        RET
;------------------------------------------------------------
                                ;SECTOR
                                ;------------------------------
                                ;searches records in sector for key(s)
                                ;starts with remaining rec in SINBUF
                                ;and reads across into INBUF
SECTOR  CALL    NEWSEC
        LD      DE,(LLRN)
        INC     DE
        LD      HL,(CLRN)
        LD      (LLRN),HL       ;CLRN will be next LLRN
        EX      DE,HL           ;is this needed??
        LD      B,A
        XOR     A               ;carry = 0
        LD      A,B             ;restore A
        SBC     HL,DE
        JP      NZ,SECLP3
        LD      A,(B256)
        CP      0DH             ;EOR?
        JP      Z,SECLP3        ;If EOR then read 1st rec
CP2E2   CP      0DH             ;'.'
        JR      Z,SECLP3        ;if EOR then read 1st rec
SECLP   CALL    RESTSC
SECLP3  CP      0
        JP      Z,ENDSEC
        CALL    SK              ;scan for keys
        LD      A,(HIT)
        CP      0               ;check if key found
        JR      Z,SECLP         ;loop until key found
SECLP4  NOP
SECLP6  LD      HL,RECBUF
        CALL    DSP             ;display record
        CP      BRKKEY          ;<break>?
        JP      Z,ENDSEC        ;RET if break
        CP      '1'             ;list cmd?
        JR      NZ,SECLP
        CALL    LIST            ;list routine
        CP      BRKKEY
        RET     Z               ;no dotted line if brk
        CALL    NEWLN
        CALL    DLINE
        CALL    NEWLN
ENDSEC  RET
;------------------------------------------------------------
                                ;NEWSEC
                                ;------------------------------
                                ;transfers logical rec from SINBUF to RECBUF
                                ;entry: NXTREC = byte in SINBUF that is start
                                ;       of next logical rec
                                ;exit:  A=0 no more records
                                ;       A=1 more logical records
                                ;       NXTREC = byte in SINBUF that is start
                                ;       of next log rec
                                ;reg altered: AF,BC,DE,HL
NEWSEC  LD      HL,SINBUF       ;next sector so NXTREC -> start of SINBUF
        LD      (NXTREC),HL
        LD      DE,RECBUF       ;HL->RECBUF
        JR      SKIP
RESTSC  LD      HL,INBUF
        LD      DE,(NXTREC)
        OR      A               ;set carry = 0
        SBC     HL,DE
        JP      M,NOMORE        ;no more rec in pt'd sec
        JP      Z,NOMORE
        LD      HL,(NXTREC)
        LD      DE,RECBUF
;
SKIP    LD      BC,0            ;skip remainder of char in record
NWSCLP  LD      A,(HL)          ;NeW SeCtor LooP
        CP      0DH
        JP      Z,NWSCE0
CP2E4   CP      0DH             ;'.' ? (pseudo EOR mark)
        JR      Z,NWSCE0        ;exit loop if EOR
        LD      (DE),A          ;transfer char from SINBUF to RECBUF
        INC     BC              ;inc char count (# of char moved to RECBUF)
        LD      A,254
        CP      C               ;record length max reached?
        JR      Z,NWSCE0        ;EOR if record >= 255 char
        INC     DE              ;pt to next char in RECBUF
        INC     HL              ;pt to next char in SINBUF
        JR      NWSCLP          ;loop until EOR found
;
NWSCE0  CP      0DH             ;end of record
        JP      Z,NWSCE4
        CP      254
        JR      Z,NWSCE4
        LD      A,(PSEOR)
        CP      0DH
        JR      Z,NWSCE4
        LD      A,'.'
        LD      (DE),A
        INC     DE
NWSCE4  LD      A,0DH
        LD      (DE),A
        INC     HL
        LD      (NXTREC),HL
        LD      A,1             ;1 -> more record  SHOULD THIS BE 1BH???
        RET
NOMORE  LD      A,0             ;0 -> no more record
        RET
;------------------------------------------------------------
                                ;SK
                                ;------------------------------
                                ;search rec in RECBUF for key(s)
                                ;entry: no para
                                ;exit:  HIT=1 if key(s) found
                                ;       HIT=0 if key(s) not found
                                ;reg altered: AF,DE,HL
                                ;calls: GTKYWD,INSTR,SKTOOR
SK      LD      HL,KEYBUF
        LD      (NXTKEY),HL     ;(NXTKEY) -> start KEYBUF
        LD      A,1
        LD      (HIT),A         ;init (HIT) = 1
        LD      A,0
        LD      (ENDKEY),A      ;init (ENDKEY)=0
        LD      (OPER),A        ;init (OPER) 0=and
        LD      (NXTOP),A       ;init (NXTOP) 0=and
;
SK2     CALL    GTKYWD          ;get 1 key word from search key
        CALL    INSTR           ;search for 1 key word
        LD      D,A             ;temp store result in D
        LD      A,(OPER)        ;OPER stores type of oper
        CP      1               ;or op?
        JP      Z,SKOR
SKAND   LD      A,(HIT)         ;and operation
        AND     D               ;D = (HIT) from INSTR
        LD      (HIT),A         ;(HIT) updated
        JP      SK4
SKOR    LD      A,(HIT)
        OR      D               ;D = (HIT) from INSTR
        LD      (HIT),A         ;(HIT) updated
SK4     LD      A,(NXTOP)       ;
        LD      (OPER),A        ;op bumped
        LD      A,(ENDKEY)
        CP      1               ;end of keys?
        JP      Z,SKEND         ;jp if end of keys
        LD      A,(HIT)
        CP      1               ;is HIT still true?
        JR      NZ,SK7
        LD      A,(NXTOP)       ;HIT=1
        CP      1               ;is next (OPER) = or?
        JR      Z,SKEND         ;if so jp to end
        JP      SK2             ;else check next key
;............................................................
                                ;SK7
                                ;..............................
                                ;HIT = 0 (search key not found)
                                ;skip to next "or"
                                ;operator in search key or
                                ;end of search key
SK7     LD      A,(OPER)        ;HIT=0; is next op = or?
        CP      1               ;is next op = "or"?
        JP      Z,SK2           ;if "or" continue search
        CALL    SKTOOR          ;skip until "or" op or
                                ;end of search key found
        CP      0               ;end of search key?
        JR      Z,SKEND         ;if end of search key jp to SKEND
        JP      SK2             ;loop until end of search key
SKEND   RET                     ;end of record search
                                ;A=0 no or; A=1 or found
                                ;(NXTKEY) set if A=1
;------------------------------------------------------------
```

Listing 9 continued

✓ See List of Advertisers on Page 291

Listing 9 continued

```
                                 ;GTKYWD
                                 ;------------------------------
                                 ;get next key word from search key
                                 ;entry: HXTKEY->start of next key word
                                 ;exit:  ENDKEY=1 if end of search key
                                 ;       ENDKEY=0 if more key words in
                                 ;                search key
                                 ;reg altered: AF,DE,HL
GTKYWD  LD      HL,(NXTKEY)      ;HL -> next key to read
        LD      DE,KEY           ;DE -> word buffer
NTCHAR  LD      A,(HL)
        CP      0DH              ;end of keys?
        JP      Z,EOKEYS
        CP      253
        JP      Z,ANDEND
        CP      254              ;OR statement?
        JP      Z,OREND
;
        LD      (DE),A           ;transfer key to word buf
        INC     HL
        INC     DE
        JR      NTCHAR           ;loop until end of keyword
;...
ANDEND  INC     HL               ;HL->next keyword or end of search key
        LD      (NXTKEY),HL
        LD      A,0DH
        LD      (DE),A           ;EOR mark
        LD      A,0
        LD      (NXTOP),A        ;load with 0 = and op
        LD      A,1
        RET
OREND   INC     HL               ;HL->next keyword or end of search key
        LD      (NXTKEY),HL
        LD      A,0DH
        LD      (DE),A           ;EOR mark
        LD      A,1
        LD      (NXTOP),A
        RET
EOKEYS  LD      (DE),A           ;place 0D at end of key
        LD      A,1
        LD      (ENDKEY),A               ;(ENDKEY)=0: end of keys
        RET
;------------------------------------------------------------
                                 ;SKTOOR
                                 ;------------------------------
                                 ;skip to "or" operator
                                 ;bypasses keywords that are "anded"
                                 ;entry:
                                 ;exit:  NXTKEY->next key word in search key
                                 ;       OPER=1 if "or" op found
                                 ;       OPER=0 if end of search key
                                 ;exit points: SKT6,SKT8
                                 ;reg altered: AF,DE,HL
                                 ;calls: none
SKTOOR  LD      HL,(NXTKEY)      ;HL->start of next key word
SKT2    LD      A,(HL)
        CP      0DH              ;end of search key?
        JR      Z,SKT4           ;jp to STK4 if EOR
        CP      254              ;'or'? operator?
        JR      Z,SKT6           ;jp to SKT6 if "or"
        INC     HL               ;pt to next char in keyword
        JR      SKT2             ;loop until EOR or or op found
SKT4    LD      A,0
SKT5    RET                      ;A=0 => EOR found
SKT6    LD      A,1              ;1 -> OR found
        INC     HL               ;have HL -> next key
        LD      (NXTKEY),HL
        LD      A,1              ;1 = or
        LD      (OPER),A         ;load (OPER) = or
SKT8    RET                      ;A=1 => "or" op
;------------------------------------------------------------
                                 ;INSTR
                                 ;------------------------------
                                 ;search RECORD for key
                                 ;entry: no parm passed
                                 ;exit:  A=1 keyword found
                                 ;       A=0 keyword not found
                                 ;reg altered: AF,BC,DE,HL
INSTR   LD      HL,RECBUF
        PUSH    HL               ;NOTE!! must restore!!!!
        LD      DE,KEY           ;DE->key word to search for
        LD      C,0
MLP     NOP                      ;main subroutine loop
        LD      A,(DE)
        LD      B,A
        LD      A,(HL)
        CP      91
        JP      P,COK
        JP      Z,COK            ;;needed??
        CP      65
        JP      M,NOTUC
        OR      32
        JP      COK
NOTUC   CP      0DH              ;end of string?
        JP      Z,NOFIND
COK     CP      B                ;cp to key char
        JP      Z,CMATCH
        JP      RESET
CMATCH  INC     C                ;char match
        INC     HL
        INC     DE
        LD      A,(DE)
        CP      0DH
        JP      Z,FOUND
        JP      MLP
RESET   LD      DE,KEY
        POP     HL
        INC     HL
        PUSH    HL
        LD      C,0
        JP      MLP
FOUND   POP     HL
        LD      A,1              ;A = 1 => print rec
        RET
NOFIND  POP     HL
        LD      A,0              ;A =0 => do not print rec
        RET
;------------------------------------------------------------
                                 ;READ2
                                 ;------------------------------
                                 ;reads 2 sectors from src file
                                 ;necessary to read 2 since record may
                                 ;span sector
                                 ;entry: BC-> 1st sector to read
                                 ;exit:
READ2   LD      DE,INDCB
READ2B  CALL    READIN           ;get first sector
        PUSH    BC
        LD      HL,INBUF
        LD      DE,SINBUF
        LD      BC,256
        LDIR                     ;transfer to first buf
;
        LD      A,(EOF)
        CP      1
        JR      NZ,READ2D
        LD      HL,INBUF
        CALL    RECBL            ;EOF so blank out INBUF
        LD      HL,INBUF
        LD      (HL),0DH         ;
        POP     BC
        RET                      ;
;
READ2D  POP     BC
        INC     BC               ;pt to next record
        CALL    REDIN0           ;get second sector
        RET
;------------------------------------------------------------
REDIN0  LD      DE,INDCB         ;skip DSPSNR sub
        CALL    READDR
        PUSH    AF
        JP      Z,REDOK2
        CP      28
        JP      NZ,ERROR
        JR      REDOK2
;
READIN  LD      DE,INDCB
        CALL    READDR
REDIN5  PUSH    AF
        JR      Z,REDOK
        CP      28
        JP      NZ,ERROR
REDOK   CALL    DSPSNR
REDOK2  LD      HL,(IDCBCR)
        LD      DE,(INLLR)
        XOR     A
        SBC     HL,DE
        JP      NZ,READRT
;
        LD      A,(IDCBOS)               ;last sector routine
        CP      0
        JP      Z,REDIN8
        LD      HL,INBUF
        LD      D,0
        LD      E,A
        ADD     HL,DE
        LD      E,A
        LD      A,255
        SUB     E
        INC     A                ;?
        LD      E,A              ;no. bytes to blank out
REDIN7  LD      A,' '
        LD      (HL),A
        LD      A,E
        CP      0
        JP      Z,REDIN8
        DEC     E
        INC     HL
        JP      REDIN7
REDIN8  LD      HL,INBUFU
        LD      (HL),0DH
        DEC     HL
        LD      (HL),0DH
;
        LD      A,1                      ;EOF reached
        LD      (EOF),A
READRT  POP     AF
        RET
;------------------------------------------------------------
                                 ;RDMSEC
                                 ;------------------------------
                                 ;read MAP sector
                                 ;entry:
                                 ;exit:
RDMSEC  PUSH    BC
        PUSH    HL
        POP     BC               ;BC now = HL
        LD      DE,MAPDCB
        CALL    READDR
        POP     BC               ;restore original BC
        RET                      ;TEMP TEMP TEMP FOR MOD III
        RET     Z
        PUSH    AF
        LD      HL,RDMSG
        CALL    VDLINE
        POP     AF
        JP      ERROR


RDMSG   DEFM    'Error while reading MAP index file'
        DEFB    0DH
;------------------------------------------------------------
                                 ;OUTREC
                                 ;------------------------------
                                 ;output record
OUTREC  LD      A,(OFSTAT)
        CP      'C'
        RET     Z                ;return if file closed
        LD      (TBC),BC
        LD      (TDE),DE
        LD      (THL),HL
        CALL    CURPOS
        LD      (CURSOR),BC
        LD      B,0
        LD      C,55
        CALL    POSCUR
        LD      HL,OUTMSG
        CALL    VDLINE
        LD      BC,(CURSOR)
        CALL    POSCUR
        LD      HL,RECBUF
OUTR4   PUSH    HL
        LD      A,(HL)
        LD      (OUTCHR),A
        LD      HL,OUTCHR
        LD      DE,OUTDCB
        CALL    WRITNX
        JR      NZ,OUTR6         ;jp to error msg
        POP     HL
        LD      A,(HL)
        CP      0DH
        JP      Z,OUTR7
        INC     HL
        JR      OUTR4
;
OUTR6   POP     HL               ;error section
        CALL    NEWLN
        CALL    ERRDSP
        CALL    NEWLN
        CALL    TAPENT
        LD      DE,OUTDCB
        CALL    CLOSE            ;close file so program may continue
        LD      A,'M'
        LD      (OMODE),A
        LD      A,'C'
        LD      (OFSTAT),A
OUTR7   NOP
        CALL    CURPOS
        LD      (CURSOR),BC
        LD      B,0
        LD      C,55
        CALL    POSCUR
        CALL    ERASEL
        LD      BC,(CURSOR)
        CALL    POSCUR
        LD      BC,(TBC)
        LD      DE,(TDE)
        LD      HL,(THL)
        RET

OUTMSG  DEFM    'Outputing'
        DEFB    3
TBC     DEFW    0
TDE     DEFW    0
THL     DEFW    0
;------------------------------------------------------------
                                 ;BACKSP
                                 ;------------------------------
                                 ;backsp 1 sector in source file
BACKSP  PUSH    DE
        PUSH    HL
        LD      HL,(IDCBCR)
        LD      A,L
        CP      0
        JP      NZ,FPOSNZ
        CP      H
        JP      Z,FPOSZ
FPOSNZ  NOP
        DEC     HL
        LD      (IDCBCR),HL
FPOSZ   POP     HL
```

Listing 9 continued

SUPER FAST!

Z80 DISASSEMBLER

$69.95

- Two pass operation - generates labels at referenced locations
- Generates Zilog mnemonics
- Allows user defined labels
- Allows define byte, define word and define space directives
- COMPLETE cross-reference
- 28 page manual
- Output to console, list or disk device(s) in any combination
- Generates mnemonics for CP/M system calls
- Illegal instructions generate define byte sequence
- Start and stop at any location in file
- Source or complete listing type output

SPEED - disassembles a typical 17K .COM file, generating a 110K .Z80 file (over 10,000 lines of source) and a 52K .XRF file in less than 1 minute 45 seconds using standard bios and 8" SS/SD!

Available for Z80 CP/M and TRS-80 III

SLR Systems

1622 North Main Street, Butler, PA 16001

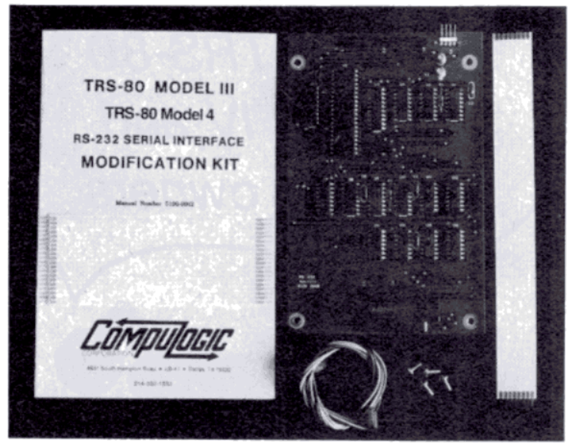
(412) 282-0864 ✓340

Terms: add $2 shipping US, others $5. PA add 6% sales tax. Specify format required. Check, MO, Visa, M/C, COD accepted

Z80, CP/M, TRS-80 TM's of Zilog, Digital Research, Tandy Corp resp.

THE RS-232

MODEL III **MODEL 4**



State of the art technology in board design, our direct replacement of Radio Shack's® internal RS-232 board, mounts inside the Model III or 4 on the existing brackets. All cables, screws and complete mounting instructions are included. Non-technical people will find that installation is quick, straight forward and simple requiring less than 15 minutes to complete.

Total compatability with Radio Shack® and all existing software is maintained. Software programmable baud rates from 50 to 19,200 baud are supported along with programmable word length, stop bits, and parity. May be utilized in either half of full duplex operation.

Outstanding Value

At Only $69.95

Guaranteed One Full Year.

Dealer Inquiries invited.

Please forward payment by a cashier check or money order.

Visa or Mastercharge also accepted.

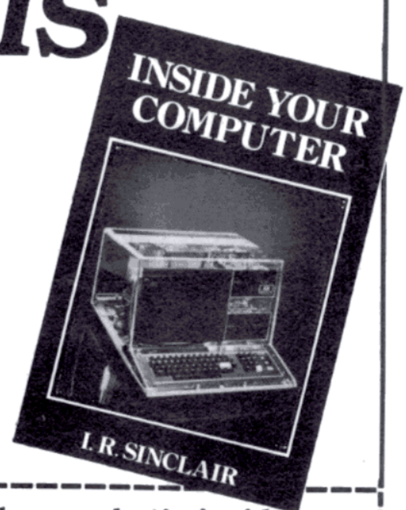
Add $3.00 shipping & handling

(Foreign orders quoted on request)

COMPULOGIC CORPORATION

4831 SOUTH HAMPTON RD. · LB41
DALLAS, TEXAS 75232
214/330-1332 ✓86

WHAT REALLY IS INSIDE YOUR COMPUTER?



Find out in **INSIDE YOUR COMPUTER** from Wayne Green Books. I.R. Sinclair takes the cover off your computer and shows you what's inside and what it does. Novices will find information on:

- Microprocessors
- Interpreters
- Registers
- Input/output
- Machine language
- Logic operations

A look at programming ties it all together—how hardware and software make a microcomputer work. The information applies to any microcomputer system. A glossary of computer terms and an appendix on binary, decimal, and hexadecimal conversion make the book all the more valuable.

$12.97, softcover, 109 pp., 5½ × 8½. ISBN #0-88006-058-1

Call **TOLL FREE 1-800-258-5473** for credit card orders. Or mail your order with payment or complete credit card information. Include $1.50 for shipping and handling. Photocopy of coupon is acceptable for ordering.

Send to:
Wayne Green Inc.
Attn: Book Sales
Peterborough, NH 03458
Dealer Inquiries Invited

Yes, I want to know what's inside my computer!

Send me ____ copies of INSIDE YOUR COMPUTER. (BK7390) Enclosed is $12.97 per copy plus $1.50 shipping and handling.

☐ MASTERCARD bank #______________ ☐VISA ☐AMEX

Card #______________ Expires______

Signature______________

Name______________

Address______________

City______________ State and Zip______________

Send To: WAYNE GREEN BOOKS Attn. Book Sales Peterborough, N.H. 03458

UPS Delivery if complete street address is given. 33DB8I

✓ See List of Advertisers on Page 291

Listing 9 continued

```
                CALL    DSPSNR
                POP     DE
                RET
;-----------------------------------------------------------------
                                            ;SEAMSG
                                            ;-------------------------
                                            ;display 'Searching' msg
SEAMSG          PUSH    DE
                PUSH    HL
                CALL    HOME                ;rl:cl
                LD      HL,MSGSEA           ;'Searching'03
                CALL    VDLINE
                CALL    NEWLN
                POP     HL
                POP     DE
                RET
;-----------------------------------------------------------------
                                            ;MENU02
                                            ;-------------------------
                                            ;mini-menu routine
MENU02          CALL    HOME                ;rl:cl
MENU02B         LD      HL,MSGM2
                CALL    VDLINE
                CALL    KBWAIT
                RES     5,A
                CP      'C'
                RET     Z
                CP      'O'
                JP      Z,M2OPT
                CP      'T'
                JP      Z,M2CMD
                CP      'M'
                JP      Z,M2CMD
                LD      A,'O'
                RET                         ;return to Enter Key
M2CMD           PUSH    AF
                LD      A,(OFSTAT)
                CP      'C'
                JP      Z,M2CMD3
                LD      DE,OUTDCB
                CALL    CLOSE               ;close output file
M2CMD3          CALL    VDINIT
                CALL    VDCLS
                POP     AF
                CP      'T'
                RET     Z
                LD      HL,MAPCMD
                RET                         ;A = 'M' so set up buffer
M2OPT           NOP
                CALL    OPT                 ;options
                LD      A,'O'
                RET
;-----------------------------------------------------------------
OPNOUT          NOP                 ;open output file
                LD      A,(OFSTAT)
                CP      'O'
                RET     Z
                LD      DE,OUTDCB
                LD      HL,OUTBUF
                LD      B,1
                CALL    OPINIT
                JP      Z,OPOUT4
                PUSH    AF                  ;error in opening file
                LD      B,PENULN
                LD      C,0
                CALL    POSCUR              ;error msg on penultimate line
                POP     AF
                CALL    ERRDSP
                CALL    BOTTOM              ;tap ent msg on bottom line
                CALL    TAPENT
                LD      A,'C'
                LD      (OFSTAT),A
                RET
OPOUT4          LD      A,'O'
                LD      (OFSTAT),A
                RET
;-----------------------------------------------------------------
PARFOR          NOP                         ;paragraph format


                LD      A,2EH
                LD      HL,CP2E2
                INC     HL
                LD      (HL),A
                LD      HL,CP2E4
                INC     HL
                LD      (HL),A
                LD      (PSEOR),A
                LD      A,'L'


                LD      (SDMODE),A                  ;display mode = line
                LD      HL,MSGPF
                CALL    VDLINE
                RET
;-----------------------------------------------------------------
RECFOR          NOP                         ;record format
                LD      A,0DH
                LD      HL,CP2E2
                INC     HL


                LD      (HL),A
                LD      HL,CP2E4
                INC     HL
                LD      (HL),A
                LD      (PSEOR),A
                LD      A,'R'
                LD      (SDMODE),A                  ;display mode = record
                RET
;-----------------------------------------------------------------


DSPSNR          NOP                         ;display sector number
                PUSH    BC
                PUSH    DE
                PUSH    HL
                PUSH    IX
                PUSH    IY
                CALL    CURPOS
                LD      (CURSOR),BC         ;save current cursor pos
                LD      BC,11
                CALL    POSCUR              ;rl:cl2
                LD      HL,(IDCBCR)         ;LRL
                CALL    DSPNBR              ;display no.


                LD      BC,(CURSOR)
                CALL    POSCUR              ;restore cursor
                POP     IY
                POP     IX
                POP     HL
                POP     DE
                POP     BC
                RET
;-----------------------------------------------------------------


LSTSEC          DEFW    0                   ;last sec: no new storage
                                            ;beyond this point
CMSEC           DEFW    0                   ;current MAP sector
NXTMAP          DEFW    0                   ;next map page
MAXMAP          DEFW    0                   ;last map byte
TBYTE           DEFB    0                   ;total hash byte
TTBYTE          DEFB    0                   ;temp total hash byte
WDLEN           DEFB    0                   ;hash word len
HIT             DEFB    0                   ;1 -> key(s) found
```

Listing 9 continued

```
NXTKEY  DEFW    0               ;next key in KEYBUF
OPER    DEFB    0               ;operation 0=and; 1=or
NXTOP   DEFB    0               ;next operation
ENDKEY  DEFB    0               ;1=end of keys reached

CLRN    DEFW    0               ;current LRN
LLRN    DEFW    0               ;last LRN
SPWORD  DEFB    0               ;(S)pace or (W)ord
EOF     DEFB    0
MODE    DEFB    0               ;list or search mode
SDMODE  DEFB    0               ;search display mode: line,rec,all
        DEFB    0               ;EXTRA
PRMODE  DEFB    'M'             ;output mode (C)ontinuous or (M)anual
OFCOND  DEFB    'C'             ;output file: (C)lose, (R)eset or (A)ppend
OFSTAT  DEFB    'C'             ;output file status: (C)losed or (O)pen

OMODE   DEFB    'M'             ;output mode (C)ontinuous or (M)anual
B256    DEFB    0               ;storage last byte SINBUF
LINENR  DEFB    0               ;line number for DSP
NINSET  DEFB    2               ;no. of spaces to inset
SINSET  DEFB    0               ;storage for inset
PSEOR   DEFB    0DH             ;pseudo EOR
;:::
CFILE   DEFS    9               ;current file name xxxxxxx3
NXTREC  DEFW    0               ;abs position of next rec
SINBUF  DEFS    256

INBUF   DEFS    255             ;second source buf
INBUFE  DEFB    0               ;byte 256 of INBUF
INBUFU  DEFB    0DH             ;end of record mark
RECBUF  DEFS    256
        DEFB    0DH             ;EOR mark
MAPBUF  DEFS    256
        DEFB    0DH
OUTBUF  DEFS    256             ;output buffer
        DEFB    0DH
OFNAME  DEFB    0DH             ;output file name

        DEFM    '
        DEFB    0DH
HSHBUF  DEFS    80
KEYBUF  DEFS    80
        DEFM    ' '             ;extra space if needed!!
        DEFB    0DH
KEY     DEFS    80
        DEFB    0DH
MSGPF   DEFM    'Paragraph Format'
        DEFB    0DH
MSGRF   DEFM    'Record Format'
        DEFB    0DH
MSGOP   DEFM    'FILES OPENED'
        DEFB    0DH

MSGEOF  DEFM    'END OF FILE'
        DEFB    0DH
MAPCMD  DEFM    'MAP'
        DEFB    0DH
CODE    DEFM    'MINDEX 1.2'
        DEFB    0DH
TITLE   DEFM    'MAP SEARCH PROGRAM  1.2'
        DEFB    3
CPYMSG  DEFM    '(c) Copyright 1983 Softshell'
        DEFB    3

THREE   DEFM    'Each key must contain at least three '
        DEFM    'characters'
        DEFB    0DH
MSGNAM  DEFM    'File Name: '
        DEFB    3
FNEMSG  DEFM    'Error in file name format'
        DEFB    0DH
MSGM2   DEFM    '(C)hange File, (M)AP, (O)ptions or (T)RSDOS: '
        DEFB    03
MAPRD   DEFM    'MAP READ COMPLETE'

        DEFB    0DH
MSGKEY  DEFM    'Enter Key '
        DEFB    3
MSGSEA  DEFM    'Searching '
        DEFB    3
QINSET  DEFM    'Inset    '
        DEFB    3
CURSOR  DEFW    0               ;cursor position
;-----------------------------------------------------------------
                                ;buffers used by S3

                                ;------------------------------
                                ;general purpose variables
ANSWER  DEFB    0               ;one byte key in variable (Y/N etc)
CMDLFS  DEFB    0               ;cmd line file spec (Y/N)
                                ;indicates whether file spec
                                ;was entered along with MINDEX cmd
                                ;hash routine variables
FBYTE   DEFW    0               ;hash value of first byte of triplet
MBYTE   DEFW    0               ;hash value of middle byte of triplet
LBYTE   DEFW    0               ;hash value of last byte of triplet

SCROSS  DEFB    0               ;sector cross variable
                                ;if 0 then sector boundary is not
                                ;crossed by a record
                                ;if >0 then the value in SCROSS
                                ;is equal to the no of bytes in record
                                ;prior to end of sector
BICT    DEFB    1               ;bit points to sector (0 to 7)
;-----------------------------------------------------------------
                                ;HEADER buffer
                                ;------------------------------

                                ;byte  label   purpose
                                ; 0- 9 HSEC    'MINDEX' and version
                                ;10-10         0DH
                                ;11-11 HSECPF  format: 0DH record
                                ;              '.' paragraph
                                ;12-19         blank (for future use)
                                ;20-43 HFN     source file spec
;-----------------------------------------------------------------
        END     BEGIN
```

Program Listing 10. Assembly-language listing of S30OPEN/SRC.

```
Ln #          Source Line

;S3OPEN/SRC   VERSION 1.2
;Jun  9, 1983
;-----------------------------------------------------------------
;SEARCH INFORMATION RETRIEVAL PROGRAM
;ROUTINE TO OPEN FILES
;COPYRIGHT 1983, SOFTSHELL CORPORATION   ALL RIGHTS RESERVED
;-----------------------------------------------------------------
S3OPEN  PSECT                           ;begin relocatable program section
```

Listing 10 continued

Listing 10 continued

```
;------------------------------------------------------------
        PUBLIC  GETFILE                 ;gets file specs
;------------------------------------------------------------
        EXTERN  CFILE                   ;current file name
        EXTERN  CMDLFS                  ;cmd line file spec (Y/N)
                                        ;indicates whether file spec also entered
                                        ;along with SEARCH cmd
        EXTERN  CODE
        EXTERN  INBUF                   ;input (source) file record buffer
        EXTERN  KEYBUF                  ;keybuf used to enter MINDEX file spec
        EXTERN  MAPBUF
        EXTERN  LSTSEC
        EXTERN  MAXMAP
        EXTERN  MENU02B
        EXTERN  NXTMAP
        EXTERN  OFNAME
        EXTERN  PARFOR
        EXTERN  RECFOR
                                        ;
;------------------------------------------------------------
                                        ;calls to MODII or MODIII
                                        ;------------------------------
        EXTERN  BOTTOM                  ;position cursor to last line of screen
        EXTERN  CLOSE                   ;close file (DE->DCB)
        EXTERN  CURPOS                  ;determine cursor pos (coor:BC)
        EXTERN  DISBRK                  ;disable break (for Model II)
        EXTERN  DIVIDE                  ;divide A into HL
        EXTERN  DOSCMD                  ;jp to TRSDOS and execute command
        EXTERN  ERASEL                  ;erase to end of line
        EXTERN  ERASES                  ;erase to end of screen
        EXTERN  ERRDSP                  ;MAP error display routine
        EXTERN  ERROR                   ;TRSDOS error msg routine
        EXTERN  HOME                    ;position cursor to top line of screen
        EXTERN  INIT                    ;initialize file (create/open, pos to lst rec)
                                        ;                (DE->DCB, HL->rec buf)
        EXTERN  JP2DOS                  ;jp to TRSDOS
        EXTERN  KBCHAR                  ;get char (or null) from keyboard
        EXTERN  KBINIT                  ;initialize keyboard (Mod II)
        EXTERN  KBLIN3                  ;get line from keyboard and end with 03
        EXTERN  KBLINE                  ;get line from keyboard and end with 0DH
        EXTERN  KBWAIT                  ;wait for char from keyboard
        EXTERN  KEYIN                   ;get search key(s) form keyboard
        EXTERN  MOVELN                  ;move line from screen to buffer
                                        ;HL->buf, D=no. char, BC=cursor coor
        EXTERN  NEWLN                   ;role screen and pos cur to next to last line
        EXTERN  OPEN                    ;open file (DE->DCB, HL->record buffer)
        EXTERN  OPINIT                  ;open and initialize file
                                        ;          (DE->DCB, HL->record buffer)
        EXTERN  POSCUR                  ;position cursor (coordinates in BC)
        EXTERN  POSEOF                  ;position to end of file
        EXTERN  POSN                    ;position to record in file
        EXTERN  PRINIT                  ;initialize printer
        EXTERN  PRLINE                  ;print line (HL->buf, B=no. char, C=end char)
        EXTERN  PRSCN                   ;print entire screen
        EXTERN  PRTBLN                  ;print last line of text
        EXTERN  PRTREC                  ;print current record
        EXTERN  READDR                  ;read file: direct access
        EXTERN  SETILR                  ;set INLRR to true last record
        EXTERN  SETMLR                  ;set MAPLLR to true last record
        EXTERN  STMEND                  ;store top of memory in MEMEND
        EXTERN  VDCHAR                  ;display char in A register
        EXTERN  VDCLS                   ;clear screen
        EXTERN  VDINIT                  ;initialize video I/O (for Model II)
        EXTERN  VDLINE                  ;display line (HL->msg)
        EXTERN  WRITNX                  ;write next record to file
;............................................................
                                        ;MODII/III buffers and I/O parameters
                                        ;..............................
        EXTERN  INDCB                   ;INput (source) file DCB
        EXTERN  IDCBRL                  ;Input file DCB Record Length
        EXTERN  IDCBCR                  ;Input file DCB Current Record
        EXTERN  IDCBLR                  ;Input file DCB Last Record
        EXTERN  INLLR                   ;true last record of source file
        EXTERN  IDCBOS                  ;Input file DCB xxxxxxx
        EXTERN  MAPDCB                  ;MAP (index) file DCB
        EXTERN  MDCBRL                  ;Map file DCB Record Length
        EXTERN  MDCBCR                  ;Map file DCB Current Record
        EXTERN  MDCBLR                  ;Map file DCB Last Record
        EXTERN  MAPLLR                  ;true last record of index file
        EXTERN  MDCBOS                  ;Map file DCB xxxxxxxx
        EXTERN  MEMEND                  ;contents ->top of memory
        EXTERN  OUTCHR                  ;record buffer (1 byte) for output file
        EXTERN  OUTDCB                  ;output file DCB
        EXTERN  PRTBUF                  ;printer buffer (80 char)
        EXTERN  SORLN                   ;start of record line no.
        EXTERN  STMAP                   ;start of MAP index file (always last byte
                                        ;of SEARCH program)
                                        ;
                                        ;I/O parameters:
        EXTERN  ARROWD                  ;down arrow
        EXTERN  ARROWL                  ;left arrow
        EXTERN  ARROWR                  ;right arrow
        EXTERN  ARROWU                  ;up arrow
        EXTERN  BRKKEY                  ;break key value
        EXTERN  CPYPOS                  ;copyright notice position (52 or 36)
        EXTERN  PENULN                  ;next to last line on screen (23 or 15)
        EXTERN  SWIDTH                  ;screen width (80 or 64)
;
;------------------------------------------------------------
                                        ;calls to GR (general routines)
                                        ;------------------------------
        EXTERN  APPEND                  ;append on string to another
        EXTERN  BXDEC                   ;binary to decimal conversion
        EXTERN  DECXB                   ;decimal to binary conversion
        EXTERN  DLINE                   ;display dotted line (e.g. ...........)
        EXTERN  DSPNBR                  ;display 5-digit decimal no. (right justified)
        EXTERN  DSPNBRL                 ;display 5-digit decimal no. (left justified)
        EXTERN  EATBNK                  ;remove leading blanks
        EXTERN  FSPEC                   ;move cmd line file psec to KEYBUF
        EXTERN  GETMFN                  ;get MAP file names
        EXTERN  LINE                    ;display line (from char in A reg)
        EXTERN  PACKFS                  ;pack (compress) file spec in fixed table
        EXTERN  PARSEF                  ;parse one file spec
        EXTERN  REC0                    ;zero-out 256-byte record
        EXTERN  RECBL                   ;blank-out 256-byte record
        EXTERN  TAPENT                  ;'Tap enter to continue'
;------------------------------------------------------------
                                        ;GR buffers
                                        ;------------------------------
        EXTERN  SNAM,SEXT,SPSW,SDDR     ;source file table
        EXTERN  INAM,IEXT,IPSW,IDDR     ;index file table
        EXTERN  ONAM,OEXT,OPSW,ODDR
;
;------------------------------------------------------------
;------------------------------------------------------------
GETFILE LD      DE,MAPDCB
        CALL    CLOSE                   ;just in case file open
        LD      DE,INDCB
        CALL    CLOSE                   ;just in case file open
GETFIL  LD      BC,2*256                ;r3:cl
        CALL    POSCUR
        CALL    ERASES                  ;erase to end of screen
        LD      BC,3*256
        CALL    POSCUR                  ;r4:cl
        LD      HL,MSGNAM               ;'File Name:' msg
        CALL    VDLINE
        LD      B,PENULN
        LD      C,0
        CALL    POSCUR                  ;lst col, next to last row
        LD      HL,MSG02                ;'Enter file name, or ? for help'
        CALL    VDLINE
;
        LD      A,(CMDLFS)              ;command line file spec(s)
        CP      'Y'                     ;where file spec's given on cmd line?
        JR      NZ,GETFIL1              ;jp to GETFIL1 if not given
        LD      BC,3*256+11
        CALL    POSCUR                  ;r4:cl2
        LD      A,'N'                   ;incase file spec on cmd line was in error
                                        ;and program needs to loop back to get
                                        ;file spec in which case the CMDLFS = N
        LD      (CMDLFS),A              ;turn off auto file spec
        LD      HL,KEYBUF               ;file spec entered on cmd line
        CALL    VDLINE                  ;dsp file spec
        LD      A,0DH
        CP      A                       ;Z (set)
        LD      B,1                     ;so rec length not = 0
        JR      GETAUTO
;
;------------------------------------------------------------
                                        ;GETFIL1
                                        ;------------------------------
                                        ;file spec not entered on cmd
                                        ;or invalid file spec form cmd
                                        ;so get file spec
GETFIL1 LD      BC,3*256+11             ;r4:cl2
        CALL    POSCUR
        CALL    ERASEL                  ;erase to end of line
        LD      B,51                    ;no more than 51 char in file spec
        LD      HL,KEYBUF               ;use KEYBUF to hold file spec
        CALL    KBLINE                  ;key in file spec
GETAUTO CP      BRKKEY                  ;was break hit?
        JR      NZ,GTFIL2               ;jp to GTFIL2 if break not pressed
;
        LD      BC,2*256                ;r3:cl
        CALL    POSCUR
        CALL    ERASES                  ;erase to end of screen
        CALL    MENU02B                 ;call mini-menu
        CP      'M'                     ;(M)AP?
        JP      Z,DOSCMD                ;jp to TRSDOS and load MAP
        CP      'T'                     ;(T)RSDOS?
        JP      Z,JP2DOS                ;jp to TRSDOS
        JP      GETFIL                  ;jp back and get file name
;
GTFIL2  LD      C,B                     ;B = no. of char's keyed in with KBLINE
        LD      A,C
        CP      0
        JR      Z,GETFIL                ;null entry so loop and get name again
;------------------------------------------------------------
                                        ;GTFILE4
                                        ;------------------------------
                                        ;file spec now in KEYBUF
                                        ;now process it
GTFIL4  LD      HL,KEYBUF
        CALL    EATBNK                  ;remove leading blanks if any
        LD      A,(HL)
        CP      '?'                     ;was ? (for help) entered?
        JP      Z,MHELP                 ;jp to MHELP (minihelp screen)
        LD      HL,KEYBUF               ;pt to KEYBUF again
        CALL    GETMFN                  ;get file names
        JR      Z,FNOK                  ;jp FNOK if file spec OK
        LD      BC,8*256                ;bad file specs
        CALL    POSCUR                  ;r8:cl
        CALL    ERASES                  ;erase to end of screen
        LD      HL,FNEMSG               ;file name error msg
        CALL    VDLINE
        CALL    TAPENT                  ;'Tap enter to continue'
        JP      GETFIL                  ;loop back and get file spec's
;------------------------------------------------------------
                                        ;FNOK
                                        ;------------------------------
                                        ;file name syntax OK
FNOK    LD      HL,SNAM                 ;HL->source file name
        LD      DE,INDCB                ;DE->index file spec
        CALL    PACKFS                  ;space compress source file spec
                                        ;and move to source file DCB
        LD      HL,INAM                 ;HL->index file name
        LD      DE,MAPDCB               ;DE->index file DCB
        CALL    PACKFS                  ;space compress index file spec
                                        ;and move to index file DCB
        LD      HL,ONAM                 ;HL->output file spec
        LD      DE,OFNAME               ;DE->output file DCB
        CALL    PACKFS                  ;space compress output file spec
                                        ;and move to index file DCB
;------------------------------------------------------------
                                        ;OPMAP
                                        ;------------------------------
                                        ;open MAP index file
OPMAP   LD      BC,5*256
        CALL    POSCUR                  ;r6:cl
        CALL    ERASES                  ;erase to end of screen
        LD      HL,IDXFIL               ;'Index File:'
        CALL    VDLINE
        LD      HL,MAPDCB               ;pt to index file spec
        CALL    VDLINE                  ;dsp MAP file name
        LD      HL,MAPBUF               ;HL->index file record buffer (ifrb)
        LD      DE,MAPDCB               ;DE->index file DCB
        CALL    OPEN                    ;open index file
        JP      Z,OPMAP6                ;jp OPMAP6 if file open OK
        PUSH    AF                      ;save TRSDOS error code
        LD      HL,MAPOER               ;MAP open error message
        CALL    VDLINE
        POP     AF                      ;restore TRSDOS error code
        CALL    ERRDSP                  ;dsp error msg
        LD      B,PENULN
        LD      C,0
        CALL    POSCUR                  ;lst col, next to last row
        CALL    TAPENT                  ;Tap enter to continue
        JP      GETFILE                 ;loop back and get file spec
OPMAP6  CALL    SETMLR                  ;set MAPLLR to true last record
                                        ;in index file
;------------------------------------------------------------
                                        ;OPSRC
                                        ;------------------------------
                                        ;open source file
OPSRC   NOP
                                        ;first get header to determine which
                                        ;file to open
        LD      BC,0                    ;header (sector 0 of index file)
        LD      DE,MAPDCB               ;DE->index file DCB
        CALL    READDR                  ;read header
        JP      Z,OPSRC3                ;jp to OPSRC3 if read OK
        CALL    ERRDSP                  ;display error msg
        LD      B,PENULN
        LD      C,0
        CALL    POSCUR                  ;next to last line, first col
        CALL    TAPENT                  ;Tapent to continue
        JP      GETFILE                 ;loop back and get file spec
;
OPSRC3  NOP                             ;check for valid header
                                        ;MAPBUF has name of soruce file
        LD      HL,MAPBUF
        LD      DE,CODE                 ;'MINDEX 1.2'
        LD      BC,9                    ;no. of bytes in CODE to be compared to
                                        ;title of header
                                        ;only 'MINDEX 1.' is checked
HDRLP   LD      A,(DE)                  ;get char form CODE
        CPI                             ;cp to (HL) and inc HL
        JR      NZ,HDRERR               ;error in header if not equal
        INC     DE                      ;pt to next char in code
        JP      PE,HDRLP                ;loop until 9 char in BC checked
        JR      HDROK                   ;'MINDEX 1.' present in header
HDRERR  LD      HL,HEADER               ;header error msg
        CALL    VDLINE
        CALL    TAPENT                  ;Tapent to continue
        JP      GETFILE                 ;jp back to beginning and start again
HDROK   NOP                             ;header ok
        INC     HL                      ;HL-> byte 10 in header
        INC     HL                      ;HL-> byte 11 in header
                                        ;header byte 11 contains indexing format
                                        ;code: '.' = paragraph format
                                        ;      0DH = record format
        LD      A,(HL)                  ;get indexing format code
        PUSH    AF                      ;save
        CP      '.'                     ;para format?
        CALL    Z,PARFOR                ;set prog for paragraph format
        POP     AF                      ;restore code
        CP      '.'                     ;para format?
        CALL    NZ,RECFOR               ;display 'record format' msg
;
                                        ;header format OK
                                        ;get source file spec
        LD      HL,MAPBUF               ;HL-> byte 0 of header
```

Listing 10 continued

See List of Advertisers on Page 291

Listing 10 continued

```
                LD      DE,20           ;offset 20
                ADD     HL,DE           ;HL->start of file spec
                LD      DE,INDCB        ;DE->source file DCB
                CALL    APPEND          ;trans file spec to DCB
                                        ;
                                        ;if source file drive specified by
                                        ;user append to current file spec
                LD      HL,SDDR         ;HL->source file drive spec
                LD      A,(HL)
                CP      0DH             ;? EOR
                JR      Z,OPSRC5        ;skip if no source file dr specified
                                        ;source file drive specified so add
                                        ;':' and drive no.
                                        ;remember DE->end of src file spec in DCB
                LD      A,':'
                LD      (DE),A          ;add colon to file spec
                INC     DE              ;bump DE
                LD      A,0DH
                LD      (DE),A          ;add EOR to file spec
                LD      HL,SDDR         ;point to source file drive spec
                CALL    APPEND          ;move src file drive spec to DCB
OPSRC5          LD      HL,SRCFIL       ;'Source File:'
                CALL    VDLINE          ;
                LD      HL,INDCB        ;source file spec in INDCB
                CALL    VDLINE          ;display source file spec
;
                                        ;move source file name to CFILE buffer
                                        ;CFILE used to display file name during
                                        ;operation of program
                LD      HL,INDCB        ;HL->to source file spec
                LD      DE,CFILE        ;DE->CFILE (current file)
                LD      C,0             ;set counter to 0
OPSR5L          LD      A,C
                CP      8               ;were eight char moved to CFILE
                JR      Z,OPSRC6        ;jp to OPSRC6 if 8 char in source file name
                LD      A,(HL)          ;get char from file spec
                CP      '/'
                JR      Z,OPSRC6        ;jp if ext
                CP      ':'
                JR      Z,OPSRC6        ;jp if drive
                CP      '.'
                JR      Z,OPSRC6        ;jp if password
                LD      (DE),A          ;copy file name char to CFILE
                INC     HL              ;pt to next char in file spec
                INC     DE              ;pt to next char in CFILE
                INC     C               ;inc counter
                JR      OPSR5L          ;loop until entire file name trasnferred
;
OPSRC6          LD      A,3
                LD      (DE),A          ;put 03 at end of file name
                                        ;03 indicates end of record
                                        ;when current file name is displayed
                                        ;VDLINE will not position cursor to next line
;
                LD      HL,INBUF        ;HL->source file record buffer (sfrb)
                LD      DE,INDCB        ;DE->source file DCB
                LD      B,0             ;record len = 0
                CALL    OPEN            ;open source file
                JP      Z,OPSRC8        ;jp to OPSRC8 if not error
                PUSH    AF              ;save error code
                LD      HL,DATAER       ;'Error in data/text file' msg
                CALL    VDLINE
                POP     AF              ;restore error code
                CALL    ERRDSP          ;dsp error msg
                LD      B,PENULN
                LD      C,0
                CALL    POSCUR          ;next to last line, 1st col
                CALL    TAPENT          ;Tapent to continue
                JP      GETFILE         ;loop back and get file spec
;
OPSRC8          LD      A,0
                LD      (IDCBRL),A      ;set LRL(record length) = 256
                CALL    SETILR          ;set INLLR to true last record
                CALL    LDMAP           ;load as much of the index file
                                        ;into memory as possible
                CALL    VDCLS           ;clear screen
                RET
;---------------------------------------------------------------
                                        ;LDMAP
                                        ;------------------------------
LDMAP           LD      HL,LDIMSG       ;loads map sectors into memory
                CALL    VDLINE
                LD      HL,(MEMEND)     ;memory end
                LD      DE,512          ;safety buffer: 512 bytes
                XOR     A
                SBC     HL,DE
                LD      (LSTSEC),HL     ;no more storage beyond
                LD      HL,STMAP
                LD      (NXTMAP),HL     ;NXTMAP -> first map byte
                LD      BC,0
LDMAPL          NOP
                INC     BC
                LD      DE,MAPDCB
                CALL    READDR
                JR      Z,LDMAP3
                CP      28
                JP      NZ,ERROR
                JP      LDMAPE
LDMAP3          NOP
                PUSH    BC
                LD      HL,MAPBUF
                LD      DE,(NXTMAP)
                LD      BC,256
                LDIR
                POP     BC
;
                LD      (NXTMAP),DE
                LD      HL,(LSTSEC)
                XOR     A
                SBC     HL,DE
                JR      C,NOROOM
                JP      LDMAPL
NOROOM          NOP                     ;no more mem for MAP
                DEC     BC
                PUSH    BC
                LD      HL,LIMIT
                CALL    VDLINE
                POP     HL
                CALL    BXDEC
                CALL    VDLINE
                LD      HL,LIMIT1
                CALL    VDLINE
                LD      HL,(MDCBCR)
                CALL    BXDEC
                CALL    VDLINE
                LD      HL,LIMIT2
                CALL    VDLINE
                CALL    TAPENT
LDMAPE          NOP                     ;end of map storage
                LD      HL,(NXTMAP)
                LD      (MAXMAP),HL     ;(MAXMAP) = last map byte
                LD      HL,MAPRD
                CALL    VDLINE
                RET
;......
LDIMSG          DEFM    'Loading MAP Index'
                DEFB    0DH
LIMIT           DEFM    'Memory not sufficient to hold entire MAP Index'
                DEFB    0DH
LIMIT1          DEFM    ' of '
                DEFB    03H
LIMIT2          DEFM    ' MAP Index sectors contained in memory'
                DEFB    0DH
;---------------------------------------------------------------
                                        ;MHELP
                                        ;------------------------------
                                        ;mini-help screen
MHELP           LD      BC,5*256        ;r6:cl
                CALL    POSCUR
                CALL    ERASES
                LD      HL,HMSG2
                CALL    VDLINE
                LD      A,0DH
                CALL    VDCHAR          ;skip line
                LD      HL,HMSG4
                CALL    VDLINE
                LD      HL,HMSG6
                CALL    VDLINE
                LD      HL,HMSG8
                CALL    VDLINE
                LD      B,PENULN
                LD      C,0
                CALL    POSCUR
                CALL    TAPENT
                JP      GETFIL1
HMSG2           DEFM    'Format:  SOURCE FILE,INDEX FILE,OUTPUT FILE'
                DEFB    0DH
HMSG4           DEFM    'Only the source file specification is required'
                DEFB    0DH
HMSG6           DEFM    'The index file will have the same name, but extension "MAP"'
                DEFB    0DH
HMSG8           DEFM    'The output file may be specified from within the program'
                DEFB    0DH
;---------------------------------------------------------------
MSG02           DEFM    'Enter file name, or ? for help'
                DEFB    3
MSGPF           DEFM    'Paragraph Format'
                DEFB    0DH
MSGRF           DEFM    'Record Format'
                DEFB    0DH
MSGOP           DEFM    'FILES OPENED'
                DEFB    0DH
MAPCMD          DEFM    'MAP'
                DEFB    0DH
HEADER          DEFM    'Error in MAP Index File Header'
                DEFB    0DH
MAPOER          DEFM    'Error in Attempt to Open MAP Index File'
                DEFB    0DH
DATAER          DEFM    'Error in Attempt to Open Data/Text File '
                DEFB    0DH
IDXFIL          DEFM    'Index File: '
                DEFB    03
SRCFIL          DEFM    'Data/Text File: '
                DEFB    03
MSGNAM          DEFM    'File Name: '
                DEFB    3
FNEMSG          DEFM    'Error in file name format'
                DEFB    0DH
MAPRD           DEFM    'MAP READ COMPLETE'
                DEFB    0DH
;---------------------------------------------------------------
                END
```

Program Listing 11. Assembly-language listing of S3DSP/SRC.

```
;S3DSP/SRC. Ver. 1.2
;Jun 21, 1983
;---------------------------------------------------------------
;SEARCH DISPLAY ROUTINE (DISPLAY RECORD OR LIST FILE)
;COPYRIGHT 1983, SOFTSHELL CORPORATION    ALL RIGHTS RESERVED
;---------------------------------------------------------------
S3DSP   PSECT                           ;begin relocatable program section
;---------------------------------------------------------------
                                        ;public routines
                                        ;------------------------------
        PUBLIC  BYUPDN                  ;bypass updown control buffer
        PUBLIC  LIST                    ;list file routine
        PUBLIC  DSP                     ;display record routine
;---------------------------------------------------------------
                                        ;calls to MODII or MODIII
                                        ;------------------------------
        EXTERN  KBLINE,KBLIN3,KBCHAR,KBWAIT,KEYIN,KBINIT
        EXTERN  KBBRK
        EXTERN  VDCLS,VDLINE,VDCHAR,VDINIT
        EXTERN  HOME,BOTTOM,NEWLN
        EXTERN  CURPOS,POSCUR
        EXTERN  SCRPRO
        EXTERN  ERASEL,ERASES
        EXTERN  OUTCHR
        EXTERN  PRINIT,PRSCN,PRTBLN,PRTREC
        EXTERN  PRLINE,MOVELN,PRTBUF
        EXTERN  POSEOF,POSN,READDR,WRITNX,BACKSP
;---------------------------------------------------------------
                                        ;MODII/III buffers and I/O parameters
                                        ;------------------------------
        EXTERN  INDCB,IDCBRL,IDCBCR,IDCBLR,INLLR,IDCBOS
        EXTERN  MAPDCB,MDCBRL,MDCBCR,MDCBLR,MAPLLR,MDCBOS
        EXTERN  OUTDCB
        EXTERN  ERRDSP
        EXTERN  SORLN,STMEND,MEMEND
        EXTERN  ARROWD,ARROWL,ARROWR,ARROWU,BRKKEY,SWIDTH,PENULN
;---------------------------------------------------------------
                                        ;S3 buffers
                                        ;------------------------------
        EXTERN  INBUF,INBUFE,INBUFU,NINSET,SINSET
        EXTERN  LINENR,SPWORD,EOF,MSGEOF,SEAMSG
        EXTERN  OMODE,SDMODE,PRMODE,MODE,RECBUF
        EXTERN  NXTREC
;---------------------------------------------------------------
                                        ;calls to S3 routines
                                        ;------------------------------
        EXTERN  READIN,OPNOUT,OUTREC
;---------------------------------------------------------------
                                        ;calls to GR (general routines)
                                        ;------------------------------
        EXTERN  BXDEC,DECXB,DSPNBR
        EXTERN  DLINE
;---------------------------------------------------------------
                                        ;LIST
                                        ;------------------------------
                                        ;list file from start of current rec
                                        ;entry: current record in RECBUF
                                        ;       current sector in INBUF
                                        ;       IX -> current src sector
                                        ;exit:
                                        ;reg altered: AF,BC,DE,HL
                                        ;calls: LIST1
LIST    LD      A,'N'
        LD      (BYUPDN),A              ;N = do not bypass updown routine
        CALL    DLINE                   ;display dotted line under last search record
        LD      A,'l'
        LD      (MODE),A                ;list mode
        PUSH    IX                      ;save
        PUSH    IY                      ;save
        PUSH    IX
        POP     BC                      ;BC ->current src sector
        LD      HL,(NXTREC)
        CALL    LIST1
        POP     IY
        POP     IX
        RET
;---------------------------------------------------------------
                                        ;LIST0
                                        ;------------------------------
                                        ;entry pts: LIST1
LIST0   LD      HL,INBUF                ;HL->src file input buffer
        JR      LIST2
LIST1   PUSH    HL                      ;save HL (-> INBUF)
        CALL    HOME                    ;r1:cl
        LD      HL,LISTM1               ;'Listing'03
        CALL    VDLINE
        CALL    NEWLN                   ;pos to next to last line
        POP     HL                      ;HL -> INBUF
        LD      A,(NINSET)              ;get current inset value
        LD      (SINSET),A              ;save current inset value
        LD      A,0
        LD      (NINSET),A              ;set inset to 0
LIST2   LD      A,1
        LD      (LINENR),A              ;set line no. = 1
        LD      B,SWIDTH                ;B = screen width
;...............................................................
```

Listing 11 continued

Seduction...
more than a game!

TRS-80 I-III-4
48K 1 Disk

$49.95

tm
adult video games
7325½ Reseda blvd.
Suite 637, Reseda, Ca.
91335 ✓56

X-rated

3400 USER® NEWSLETTER

Word Processing/Data Base News for Model III, IV, & 100

3400 USER® is the first newsletter devoted primarily to WORD PROCESSING & DATA BASE use for Models III, IV, & 100. It is intensely reader/user orientated on the premise that average users discover outstanding time-saving ideas and want to share. We publish these submitted gems the very next month using simple low-budget printing methods. Includes monthly TUTORIAL on some aspect of word processing/data base you can use NOW, & actually see your investment produce! Corporate product news, bugs & fixes covered promptly! **3400™** is for "Average Down-to-Earth Word Processor Users" who don't know their Peek from their Poke. **3400™** is for YOU, the day-to-day user. (If that's actually your secretary . . . she'll love you for subscribing.) Join up before January 84 & receive our "JUST DO IT!" poster with your first issue. Submissions welcomed. $24.00 per year, 1st class.

Send cash, check, money order to:
CREST SOFTWARE ✓223
2132 Crestview Drive • Durango, CO 81301
(303) 247-9518
Visa, MC accepted, include card # and expiration date.
3400 USER® & 3400™ are trademarks of Crest Software, Durango, CO

wabash®

six-year warranteed ✓367

DISKETTES!
$18.95/box (10)
with FREE library case!

5¼" single-side, single-density; double-density add $2/box. 8" disks comparably priced. Add $2 per order shipping. *In Illinois add 6% sales tax.* Immediate shipment on VISA, Master Card or Money Order; Add 14 days for personal checks.

CALL TOLL FREE
(800) 222-1248
In Illinois Call (312) 882-8315

DEALERS! SCHOOLS! USER GROUPS!
Call for our **low** volume discount prices!

DIGITAL IMAGES
Box 941005, Schaumburg IL 60194

LIMITED OFFER

NEWDOS - 80 ver. 2.0
and DOSPLUS ver. 3.4

A combined regular price of
$299.96

Now $149.00 combined price

or call for separate pricing. Offer subject to cancellation without notice.

CALL TOLL FREE
800-835-0071

In Kansas
316-665-3611

SALES DATA, INC.

526 E. 4th ✓142
HUTCHINSON, KANSAS 67501

ESSJAY

CUT YOUR PAPERWORK TO shreads with SIMPLEX FORMS PROCESSORS

**** MULTI-PURPOSE CHECK WRITER ****
uses most Multi-purpose checks for Payroll, Accounts Receivable, and Cash Disbursement checks. Each type followed by SESSION REPORT for further bookkeeping procedures.
Specify Model I or III Tape or Disk. **$39.95**

**** MULTI-PURPOSE FORM PROCESSOR ****
Uses most multi-purpose printed 8-½ X 11 forms or PLAIN PAPER. Your choice.
Processes: INVOICES, STATEMENTS, PURCHASE ORDERS, MEMOS.
Specify as above. only **$49.95**

**** SPECIAL — COMBO PACK of BOTH ****
Specify as above. only **$79.97**

ALSO many single SIMPLEX PROCESSORS
NEW programs — soon — CONTRACTOR CERTIFIED PAYROLL, REG. PAYROLL, PAYABLES, RECEIVABLES

ADD $3.00 SHIPPING AND HANDLING
Tenn. Residents add 6¾ percent tax

DEALERS INVITED ✓48

ESSJAY P.O. Box 566
Dyersburg, TN 38024 -- Phone 901-285-1800

Reviewers agree: quality documentation can mean success or obscurity for your program.

FORMAN AND WILLIAMS DOCUMENTATION SERVICE

We'll evaluate your original program documentation for clarity, user-friendliness, and salability. Then, with your approval, our marketing professionals will edit, revise, or rewrite documentation to help you create a total marketing package for your Basic program.

Send your Model 1 Level II cassette or single-user Model II/12/16 8" floppy diskette, current documentation, program listing, and self-addressed, stamped envelope to

FORMAN AND **WILLIAMS**
3387 Del Rosa Ave., North (Suite 113)
San Bernardino, CA 92404 ✓451

Programs will be returned **only if accompanied** with sufficient return postage and envelope.
NOTE: We do not evaluate or debug programs

•••SPECIAL NOTE to Software Manufacturers, Computer Stores, and Mail Order Houses: send #10 SASE and your business card for information on our complete advertising and marketing services.•••

IS YOUR PRINTOUT FADING?
THEN YOU NEED:
RIBBON SAVER!
REVITALIZE YOUR FABRIC RIBBON WITH "RJ-1000B"- APPLICATOR

- Will revive all fabric cartridges, (except carbon)
- Works on all FABRIC ribbons, (all colors and sizes)
- Simple, no tools needed
- Extend life of your ribbon up to 3 times, or more
- Can also be used to rejuvenate fabric home typewriter ribbon
- Satisfaction guaranteed, or money back

PRICE: $3.95

shipping & handling included in cont. US
(Foreign: add $2.00 extra, (pay in U.S. funds)
NJ residents add 6%
For heavily used ribbons order "RP-2010B" $1.00 extra (state color)

send check or M.O. to: **DESIGN COLORANTS**
P.O. Box 612
✓202 **Wayne, NJ 07470**

For VISA or M.C. orders call: (201) 667-7545

FINALLY!!

ARCADE GAMES AND GOOD GRAPHICS FOR

MODEL 2, 12, 16

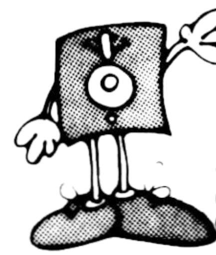
$19.95 each - or two for $29.95!

Galactic Invaders . . . . . Munchman
Quest for Adventure . . . . . Meteors
Space Trek . . . . Wormy
Computer Casino . . . . Bust-out
The Wrath of Ken . . . . Chess
Racer X . . . . Othello

Call or write for full game list
KUZEL COMPUTER SERVICES
8654 W. Berwyn Ave. #3S
Chicago, Illinois 60656
(312) 399-0273 ✓354

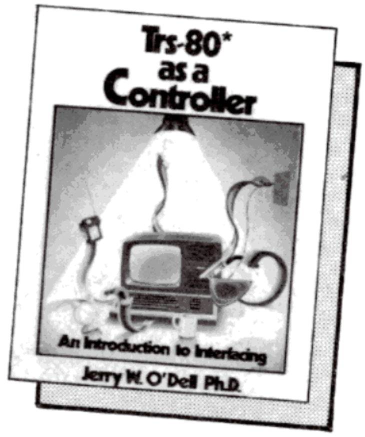
MEMOREX
FLEXIBLE DISCS

WE WILL NOT BE UNDERSOLD!! Call Free **(800)235-4137** for prices and information. Dealer inquiries invited and C.O.D.'s accepted.



PACIFIC EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA
93401 *In Cal. call*
(800)592-5935 or
(805)543-1037
✓207

PUT YOUR TRS-80* IN CONTROL



Interface your TRS-80 to outside devices. Learn with the projects in ***TRS-80 as a Controller.*** You can use your computer to control lights, switches, and even a small computer you build yourself.

All it takes is a minimum knowledge of electronics and programming. Circuits are simple. Most programs are fewer than fifteen lines long. The instructions are clear and fully illustrated with photographs, schematics, and figures.

Jerry O'Dell has designed these projects to be both easy and inexpensive. You don't need disk drives, plotters, digitizers or other fancy units.

You don't have to be an expert and you don't have to be rich.

All you need is a TRS-80 Model III with 16K RAM, Level II BASIC, and a few other parts that you will no doubt find useful at a later date. (You can also use a Model I, with the conversions provided in an appendix.) The components you'll need are all readily available.

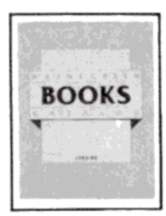
The book begins with a description of the Model III and Z80 and all the chips, circuits, prototyping boards, and other devices used in TRS-80 interfacing. There are also helpful suggestions throughout for expanding the projects into more complex applications.

Jerry W. O'Dell, Ph.D., is a psychology professor at Eastern Michigan University. He has published many articles, including several in ***80 Micro*** and the ***Encyclopedia for the TRS-80.***

BK7394 $12.97 softcover
7 by 9 approx. 176 pp.
ISBN 0-88006-061-1
Wayne Green Books 1983

Credit card orders call **TOLL-FREE 1-800-258-5473** or mail your order with payment of $12.97 each plus $1.50 per book shipping and handling to: Wayne Green Book Sales, Peterborough, NH 03458.

Dealer inquiries invited



Check the box on the coupon for your **FREE WAYNE GREEN BOOKS 1983 RETAIL CATALOG.**

Due to printer error, unavailable until November 30, 1983. We apologize for any inconvenience.

WG BOOKS

Put my TRS-80 in control. Please rush me ______ copies of **TRS-80 as a Controller** (BK7394) at $12.97 each.

☐ Enclosed is $12.97 per copy plus $1.50 for shipping and handling. ☐ Please send me a 1983 retail catalog.

☐ MASTERCARD MC bank #__________
☐ VISA ☐ AMEX
Card #__________ Expires______
Signature__________
Name__________
Address__________
City__________
State__________ Zip______
All orders shipped UPS if complete street address is given.
33DB8T

* **TRS-80 is a trademark of the Radio Shack division of Tandy Corp.**

Listing 11 continued

```
                                ;LIST3
                                ;................................
                                ;process next word
LIST3   CALL    NXTWRD          ;get next word
        LD      A,'W'
        LD      (SPWORD),A      ;working with word
        PUSH    HL              ;save HL
        LD      HL,INBUFU       ;0DH byte after INBUFE
        XOR     A
        SBC     HL,DE
        POP     HL              ;restore HL
        JP      Z,LIST40        ;get next sec
        LD      A,0
        CP      C
        JR      Z,LIST3C
        CALL    DSPWRD          ;display word
        CP      BRKKEY          ;<break> ?
        JP      Z,LIST70        ;jp LIST70 if <break>
        CP      's'             ;was 's' pressed
        JP      Z,LIST70        ;end, then search
        CP      ARROWU          ;was uparrow pressed? (backspace)
        JP      Z,LIST0
        LD      A,(HL)
        CP      0DH
        JP      Z,LIST5
LIST3C  CALL    SPACE           ;get next space or set of spaces
        LD      A,'S'
        LD      (SPWORD),A      ;space-word = (S)pace
        PUSH    HL
        LD      HL,INBUFU
        XOR     A
        SBC     HL,DE
        POP     HL
        JP      Z,LIST40
        CALL    DSPSP0          ;used for listing only
        CP      BRKKEY          ;<break>?
        JP      Z,LIST70        ;exit and get new search key
        CP      's'             ;[s] pressed?
        JP      Z,LIST70        ;exit and resume search
        CP      ARROWU          ;[up arrow]?
        JP      Z,LIST0         ;backsp, so start listing again
LIST4   LD      A,(HL)
        CP      0DH             ;EOR?
        JP      NZ,LIST3        ;loop until EOR encountered
;...............................................................
                                ;LIST5
                                ;................................
                                ;EOR so check if end of INBUF
LIST5   LD      DE,INBUFE       ;DE-> char after INBUF
        PUSH    HL
        XOR     A
        SBC     HL,DE
        POP     HL              ;restore HL
        JP      Z,LIST20        ;jp if HL at end of buffer
        CALL    UPDOWN
        CP      BRKKEY          ;<break>?
        JP      Z,LIST70        ;exit and get next search key
        CP      ARROWU          ;up arw
        JP      Z,LIST0         ;backsp, so start listing again
        CP      's'             ;[s]earch cmd
        JP      Z,LIST15        ;exit and continue search
        LD      A,SWIDTH        ;A=screen width
        CP      B               ;new line?
        JP      Z,LIST9
        CALL    NEWLN           ;roll screen in new line needed
        LD      B,SWIDTH        ;reset count for char printed on line
LIST9   INC     HL              ;HL->next char after 0D
        JP      LIST3           ;loop until EOR
LIST15  CALL    NEWLN           ;roll screen
        LD      A,'s'
        JP      LIST70
;...............................................................
                                ;LIST20
                                ;................................
                                ;0DH at INBUFE (record extends past
                                ;INBUF buffer)
LIST20  CALL    UPDOWN
        CP      BRKKEY          ;<break>?
        JP      Z,LIST70        ;exit and get new search key
        CP      ARROWU          ;[up arrow]?
        JP      Z,LIST0         ;backsp, so start list again
        CP      's'             ;[s]earch cmd?
        JP      Z,LIST15
        LD      A,SWIDTH
        CP      B               ;new line?
        JP      Z,LIST23
        CALL    NEWLN           ;roll screen
        LD      B,SWIDTH
LIST23  LD      C,0
        LD      A,'W'
        LD      (SPWORD),A
        LD      DE,INBUF
        JP      LIST48
;...............................................................
                                ;LIST40
                                ;................................
                                ;word or spaces cross sec
                                ;shift word/sp to SINBUF
LIST40  LD      A,(EOF)
        CP      1               ;has EOF been reached?
        JP      Z,LIST50        ;jp LIST50 if EOF
        LD      HL,INBUFE       ;transfer remainder of INBUF to SINBUF
        LD      DE,INBUF
        DEC     DE
        PUSH    BC
        LD      B,0
        LD      A,C
        CP      0               ;ct 0?
        JP      NZ,LIST43       ;if 0 then no trans
        POP     BC              ;restore stack
        JP      LIST48
LIST43  INC     BC              ;include char before string to be transferred
        LDDR                    ;trans to SINBUF
        INC     HL              ;adjust for inc of BC
        INC     DE              ;adjust for inc of BC
        INC     DE              ;DE-> 1st char of word
        PUSH    DE
        PUSH    HL
;...............................................................
                                ;LIST45
                                ;................................
LIST45  LD      A,(EOF)
        CP      1               ;was EOF reached?
        JP      Z,LIST50        ;jp LIST50 if EOF
        PUSH    BC
        PUSH    DE
        LD      DE,INDCB
        LD      BC,(IDCBCR)     ;get current record
        INC     BC              ;now points to next record
        CALL    READIN          ;read next record
        POP     DE
        POP     BC
LIST47  POP     HL              ;HL->INBUFE-C
        POP     DE              ;DE-> wrd start in SINBUF
        EX      DE,HL           ;HL-> wrd start in SINBUF
        POP     BC
        LD      DE,INBUF
        LD      A,(SPWORD)
        CP      'S'
        JP      Z,LIST3C
        JP      LIST3
LIST48  NOP                     ;no word/sp in SINBUF
        LD      HL,INBUF        ;start word at INBUF
        PUSH    BC
        PUSH    DE
        PUSH    HL
        JP      LIST45
LIST50  NOP                     ;EOF reached
        POP     HL
        POP     DE
        POP     BC
        CALL    NEWLN
        LD      HL,MSGEOF
        CALL    VDLINE
LIST52  CALL    KBWAIT
        CP      BRKKEY          ;<break>?
        JP      Z,LIST70
        OR      32
        CP      's'
        JP      NZ,LIST52
        JP      LIST70
;...............................................................
                                ;LIST70
                                ;................................
                                ;exit from LIST routine
LIST70  PUSH    AF              ;save A status
        LD      A,(SINSET)
        LD      (NINSET),A      ;restore inset
        LD      A,0
        LD      (EOF),A         ;restore to not EOF
        POP     AF              ;restore A
        CP      's'
        RET     NZ              ;return if <break>
        CALL    SEAMSG          ;'Searching'03
        LD      A,'s'           ;return and continue searching
        RET
;...............................................................

LISTM1  DEFM    'Listing    '
        DEFB    3
;---------------------------------------------------------------
                                ;DSP
                                ;--------------------------------
                                ;display record found while searching
                                ;record is in RECBUF
                                ;B reg used for count of char remaining
                                ;       on current line
                                ;entry:
                                ;exit:  A = cmd from UPDOWN routine
                                ;reg altered: AF,BC,DE,HL
                                ;calls: NXTWRD,DSPWRD,SPACE,DSPSP
DSP     LD      A,(OMODE)
        CP      'C'             ;is output mode continuous?
        CALL    Z,OUTREC        ;if so, output record
        LD      A,PENULN
        INC     A               ;since NEWLN dec's SORLN as soon as
                                ;record is displayed
        LD      (SORLN),A       ;start of record on penultimate line
        LD      A,(SDMODE)
        CP      'R'             ;record?
        JP      Z,DSPRC         ;display entire record
        CP      'C'             ;continuous display?
        JP      Z,DSPRC         ;display entire record
;...............................................................
                                ;DSPLN
                                ;................................
                                ;display record a line at a time
                                ;call UPDOWN after each line
DSPLN   LD      A,'N'
        LD      (BYUPDN),A      ;do (N)ot bypass UPDOWN routine
        JR      DSP1
;...............................................................
                                ;DSPRC
                                ;................................
                                ;display a entire record at one time
                                ;call UPDOWN after rec is displayed
DSPRC   LD      A,'Y'
        LD      (BYUPDN),A      ;do (Y)es bypass UPDOWN routine
;...............................................................
                                ;DSP1
                                ;................................
                                ;start display
DSP1    LD      A,'s'
        LD      (MODE),A        ;search mode
                                ;string ends with 0DH
        LD      A,1             ;1 = first line
        LD      (LINENR),A      ;2 = 2..n lines
        LD      B,SWIDTH
        JP      DSP4            ;checks for null record
DSP3    CALL    NXTWRD          ;get next word
        CALL    DSPWRD          ;display word
        CP      BRKKEY          ;<break>?
        JP      Z,DSP6          ;exit if break
        CP      'l'             ;switch to list mode?
        JP      Z,DSP8          ;exit if switch in mode
        CP      'e'             ;was last char at 80?
        JP      Z,DSP12
        LD      A,(HL)          ;pt to next char in record
        CP      0DH             ;EOR?
        JP      Z,DSP5          ;exit if EOR
        CALL    SPACE           ;get next set of spaces form record
        CALL    DSPSP           ;display spaces
        CP      BRKKEY          ;<break>?
        JP      Z,DSP6          ;exit if break
        CP      'l'             ;switch to list mode?
        JP      Z,DSP8          ;exit if switch in mode
DSP4    LD      A,(HL)          ;get next char in record
        CP      0DH             ;check for null record
        JP      NZ,DSP3         ;loop if not EOR
;...............................................................
                                ;DSP5
                                ;................................
                                ;EOR reached
DSP5    LD      A,(SDMODE)      ;get search display mode
        CP      'C'             ;?continuous
        JP      Z,DSP5A         ;jp DSP5A if continuous
        LD      A,'N'
        LD      (BYUPDN),A      ;do not bypass UPDOWN
DSP5A   CALL    UPDOWN
        PUSH    AF
        CALL    NEWLN           ;roll screen
DSP5B   POP     AF              ;restore A
DSP6    RET
;...............................................................
                                ;DSP8
                                ;................................
                                ;switch from search to list mode
DSP8    CALL    NEWLN
        LD      A,'l'           ;return with A = 'l' for list mode
        RET
;...............................................................
                                ;DSP12
                                ;................................
DSP12   LD      A,(SDMODE)
        CP      'C'
        JP      Z,DSP14
        LD      A,'N'
        LD      (BYUPDN),A      ;do not bypass UPDOWN
        CALL    UPDOWN          ;char on 80; cur at 1
        PUSH    AF
        CALL    NEWLN
        POP     AF
        RET
;...............................................................
                                ;DSP14
                                ;................................
                                ;exit: down arrow pressed
DSP14   CALL    NEWLN
        LD      A,ARROWD
        RET
;---------------------------------------------------------------
                                ;NXTWRD
                                ;--------------------------------
                                ;get next word
                                ;entry:
                                ;exit:
                                ;entry: HL -> start of word
                                ;exit:  HL -> start of word
                                ;       DE-> char after word
                                ;       C = word len
                                ;       B = # char remaining on line
                                ;       A = char after word
```

Listing 11 continued

Listing 11 continued

```
                                        ;reg altered: AF,C,DE
NXTWRD  PUSH    HL                      ;save start of word
        LD      C,0                     ;initialize char count
NWRDLP  LD      A,(HL)
        CP      0DH                     ;EOR?
        JP      Z,NWRDRT                ;EOR encountered
        CP      20H                     ;' '?
        JP      Z,NWRDRT                ;space encounterd
        LD      A,SWIDTH
        CP      C
        JP      M,NWRD65                ;don't inc C after SWIDTH
        INC     C                       ;bump char counter
NWRD65  INC     HL                      ;bump HL
        JP      NWRDLP                  ;loop until end of word
NWRDRT  LD      D,H                     ;DE-> char after word
        LD      E,L
        POP     HL                      ;HL->start of word
        RET
;-------------------------------------------------------------
                                        ;DSPWRD
                                        ;-----------------------------
DSPWRD  NOP                             ;display word
                                        ;entry: HL -> start of word
                                        ;       C = word length
                                        ;       B = # spaces left on line
                                        ;exit:  B = # spaces left on line
                                        ;exit pts: DWRD2,DWRD7,DWRD11,DWRD12
                                        ;reg altered: AF,BC,DE,HL
DWRD2   LD      A,0
        CP      C
        RET     Z                       ;return if null word
        LD      A,B
        CP      C                       ;is there room for word?
        JP      Z,DWRD4
        JP      P,DWRD4
        PUSH    DE
        CALL    UPDOWN
        CP      ARROWD                  ;? down arrow
        JP      NZ,DWRD10               ;jp if not d. arrow
        CALL    NEWLN
        POP     DE
        CALL    INSET
DWRD4   LD      A,(HL)                  ;point to char
DWRD5   AND     7FH                     ;mask bit 7
        CALL    VDCHAR                  ;display char
        DEC     B                       ;one less space
        LD      A,0
        CP      B
        JP      NZ,DWRD7
        INC     HL
        LD      A,(HL)
        CP      0DH
        JP      Z,DWRD12                ;no inset if EOR
        DEC     HL
        CALL    UPDOWN
        CP      ARROWD                  ;down arrow?
        JR      NZ,DWRD11               ;jp DWRD11 if not down arrow
        CALL    NEWLN
        CALL    INSET                   ;inset and adjust B (# char remaining on line)
DWRD7   INC     HL                      ;bump char pointer
        LD      A,(HL)                  ;get next char
        CP      ' '
        RET     Z
        CP      0DH
        RET     Z
        JP      DWRD5                   ;loop until word end
DWRD10  POP     DE
DWRD11  RET
DWRD12  LD      A,'e'                   ;last char of word on last char of line
        RET
;-------------------------------------------------------------
                                        ;SPACE
                                        ;-----------------------------
                                        ;read spaces
                                        ;entry: HL -> start sp
                                        ;exit:  HL -> start sp
                                        ;       DE -> next word
                                        ;       C = # of spaces
                                        ;reg altered: AF,C,DE
                                        ;calls: none
SPACE   PUSH    HL                      ;save start of spaces
        LD      C,0                     ;initialize space count
SPACE4  LD      A,(HL)                  ;get next char
        CP      20H                     ;' '?
        JP      NZ,SPACE7               ;jp SPACE7 if not space
        LD      A,SWIDTH
        CP      C
        JP      M,SPACE5                ;don't inc # of spaces if past end of line
        INC     C                       ;inc # of spaces
SPACE5  INC     HL                      ;pt to next char
        JP      SPACE4                  ;loop until non-space char
SPACE7  EX      DE,HL                   ;DE->next word
        POP     HL                      ;HL->start of spaces
        RET
;-------------------------------------------------------------
                                        ;DSPSP0
                                        ;-----------------------------
                                        ;DSPSP routine when listing
                                        ;see DSPSP below for more info
DSPSP0  LD      A,1
        LD      (LINENR),A
DSPSP0C PUSH    DE
        DEC     HL
        LD      A,(HL)
        INC     HL
        CP      0DH                     ;was last char before space EOR?
        JP      Z,DSPSP3
        JP      DSPSP2
;-------------------------------------------------------------
                                        ;DSPSP
                                        ;-----------------------------
                                        ;display spaces
                                        ;entry: HL->start of spaces
                                        ;       C = # of spaces
                                        ;       B = # char reamining on line
                                        ;exit:  A = UPDOWN cmd
DSPSP   PUSH    DE                      ;save start of next word
        LD      A,(LINENR)
        CP      1
        JP      Z,DSPSP3                ;skip if first line
DSPSP2  LD      A,(NINSET)
        ADD     A,B
        CP      SWIDTH                  ;start of line?
        JP      Z,DSPSP8                ;bypass if start of 2...n
DSPSP3  LD      A,B
        CP      C                       ;is there room for spaces
        JP      M,SPEOLN                ;jp to eoln routine
        JP      Z,SPEOLN
        LD      A,(HL)
DSPSP7  PUSH    DE
        CALL    VDCHAR
        POP     DE
        DEC     B
        INC     HL
        LD      A,(HL)
        CP      0DH
        JP      Z,DSPSP8
        CP      ' '
        JP      NZ,DSPSP8
        JP      DSPSP7
DSPSP8  POP     HL                      ;HL->next word
        LD      A,0
        RET
;.............................................................
                                        ;SPEOLN
                                        ;.............................
                                        ;spaces to be printed at end of line
                                        ;therefore truncate spaces
SPEOLN  CALL    UPDOWN
        CP      ARROWD                  ;? down arrow
```

Listing 11 continued

Listing 11 continued

```
        JP      NZ,DSPSP9       ;jp if not d arrow
        CALL    NEWLN
        CALL    INSET
        JR      DSPSPG
DSPSP9  PUSH    AF
        PUSH    DE
        LD      B,A
        CP      0
        JP      Z,DSPSPB
        CALL    NEWLN
DSPSPB  POP     DE
        POP     AF
DSPSPG  POP     HL              ;HL->next word
        RET
;-----------------------------------------------------------------
                                ;INSET
                                ;-------------------------------
                                ;new line so inset number of
                                ;spaces in NINSET
INSET   PUSH    BC              ;save C
        LD      B,SWIDTH        ;B=screen width
        LD      A,(NINSET)      ;no. of spaces to inset
        CP      0               ;inset = 0?
        JR      Z,INSET0        ;no inset
        LD      C,A             ;C now has inset value
INSETL  LD      A,' '           ;inset loop
        CALL    VDCHAR          ;dsp ' '
        DEC     B               ;dec no. of char remaining on line
        DEC     C               ;dec no. of char to inset
        LD      A,0
        CP      C               ;end of inset loop?
        JP      NZ.INSETL       ;loop until inset complete
INSET0  POP     DE              ;exit from routine
        LD      C,E             ;restore C
        LD      A,2
        LD      (LINENR),A
        RET
;-----------------------------------------------------------------
                                ;UPDOWN
                                ;-------------------------------
                                ;This routine is called after a line
                                ;is displayed by the LIST or DSP routines
                                ;printing is initiated from this routine
                                ;entry:
                                ;exit:  A returns key pressed
                                ;       one exit point: UPDNED
                                ;reg altered: AF,BC,HL
UPDOWN  PUSH    DE              ;save DE for entire routine
        LD      A,(PRMODE)
        CP      'C'             ;is print mode continuous?
        CALL    Z,PRTBLN        ;if so print line
        LD      A,(BYUPDN)      ;
        CP      'N'             ;is display mode manual?
        JP      Z,UPDNLP        ;continue if dsp mode manual
        LD      A,ARROWD        ;continous dsp mode
        JP      UPDNED          ;return with down arrow in A reg
;.................................................................
                                ;UPDNLP
                                ;...............................
                                ;master loop of UPDOWN routine
UPDNLP  CALL    KBINIT          ;clear key buf
        LD      A,2
        CALL    VDCHAR          ;cursor off
        CALL    KBWAIT          ;wait until key is pressed
        CP      BRKKEY          ;<break>?
        JP      Z,UPDNED        ;jp to end of routine if break
        CP      ARROWD          ;down arrow?
        JP      Z,UPDNED        ;jp to end of routine if down arrow
        CP      ARROWU          ;up arrow?
        JP      Z,UPDNU         ;jp to back space routine
        OR      32              ;convert char entered to lower case
        CP      'l'             ;'l' => switch from search to list
        JP      Z,UPDND         ;jp to list/exit section
        CP      's'             ;'s' => switch from list to search
        JP      Z,UPDNS         ;jp to search/exit section
        CP      'w'             ;'w' => write (print) entire screen
        JP      Z,UPDNW         ;jp to write screen section
        CP      'p'             ;'p' => print line
        JP      Z,UPDNP         ;jp to print section
        CP      'o'             ;'o' => output current record
        JP      Z,UPDNO         ;jp to output section
        JR      UPDNLP          ;loop until arrow or break pressed
;.................................................................
                                ;UPDND
                                ;switch from search to list mode
                                ;...............................
UPDND   LD      A,(MODE)
        CP      'l'             ;list mode?
        JP      Z,UPDNLP        ;do not exit if already in list mode
        LD      A,'l'
        LD      (MODE),A        ;set MODE=(l)ist
        JP      UPDNED          ;exit
;
;.................................................................
                                ;UPDNO
                                ;...............................
UPDNO   CALL    OUTREC          ;output record
        JP      UPDNLP          ;loop until arrow or break pressed
;
;.................................................................
                                ;UPDNP
                                ;...............................
                                ;print either a line or current record
UPDNP   LD      A,(SDMODE)      ;get search-display mode
                                ;L=line, R=record, C=continuous
        CP      'L'             ;line?
        JP      Z,UPDNPL        ;print line
        CP      'C'             ;continuous?
        JP      Z,UPDNLP        ;print line
        PUSH    BC
        PUSH    DE
        PUSH    HL
        CALL    PRTREC          ;print record
        POP     HL
        POP     DE
        POP     BC
        JR      UPDNPE          ;exit this section
UPDNPL  CALL    PRTBLN          ;print line
UPDNPE  JP      UPDNLP          ;loop until arrow or break pressed
;.................................................................
                                ;UPDNW
                                ;...............................
                                ;write (print) entire screen
UPDNW   PUSH    BC
        PUSH    DE
        PUSH    HL
        CALL    PRSCN           ;print screen
        POP     HL
        POP     DE
        POP     BC
        JP      UPDNLP          ;loop until arrow or break pressed
;.................................................................
                                ;UPDNS
                                ;...............................
                                ;switch from list to search mode
UPDNS   LD      A,(MODE)
        CP      's'             ;search mode?
        JP      Z,UPDNLP        ;do not exit if already in search mode
        LD      A,'s'
        LD      (MODE),A        ;MODE=(s)earch
        JP      UPDNED          ;exit
;.................................................................
                                ;UPDNU
                                ;...............................
                                ;up arrow press (backspace)
UPDNU   LD      A,(MODE)
        CP      'l'             ;list mode?
        JP      NZ,UPDNLP       ;do not back space if not list mode
        LD      DE,INDCB        ;DE->source file DCB
        CALL    BACKSP          ;backspace one sector
        CALL    BACKSP          ;backspace one sector
        CALL    BACKSP          ;backspace one sector
        PUSH    BC
        LD      BC,(IDCBCR)     ;BC->current record
        CALL    READIN          ;read record
        POP     BC
        CALL    NEWLN           ;roll screen one line
        LD      A,0
        LD      (EOF),A         ;not EOF
        LD      HL,MSGBAK       ;'*Backspace 3 sectors*'03
        CALL    VDLINE          ;dsp msg
        CALL    NEWLN           ;rool scrren one line
        LD      A,ARROWU        ;return with up arrow in A
;.................................................................
                                ;UPDNED
                                ;...............................
                                ;exit from UPDOWN
UPDNED  POP     DE              ;restore DE
        RET
BYUPDN  DEFB    0               ;bypass updown routine


MSGBAK  DEFM    '*Backspace 3 sectors*'
        DEFB    03
;-----------------------------------------------------------------
        END
```

READ THIS

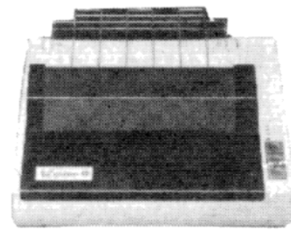
EPSON

Lowest Prices!

RX-80 FX-80
RX-80 FT FX-100
RX-100

Star Micronics GEMINI

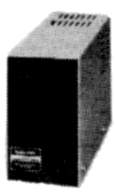
Prices So Low, We Can't Even Print Them!



GEMINI 10x
GEMINI 15x

DISK DRIVES for the TRS-80 Mod 1, 3, 4

Bare Tandon 100-1 **$169.95**
2 TEAC 55A *in Cabinet* **$399**
Call or Write for Complete Pricing



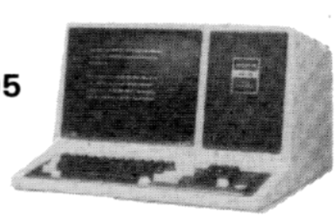
DISKETTES $19.95

CDC Brand, 5¼", Double Density

Box of 10 Diskettes

TRS-80 Mod 4

64K, 2 Drive $1595



RS-232 — $99.95

For printer prices and a FREE CATALOG, call our TOLL FREE, quick information Hot Line

800-331-3896

In Oklahoma, Call **918/825-4844**

THESE ARE OUR CASH DISCOUNTED PRICES. C.O.D. AND CHARGE ORDERS ARE 3% HIGHER. ALL COMPUTERS ARE SHIPPED FREIGHT COLLECT. PRICES, SPECIFICATIONS, AND AVAILABILITY ARE SUBJECT TO CHANGE WITHOUT NOTICE. IBM & IBM PERSONAL COMPUTERS ARE TRADEMARKS OF INTERNATIONAL BUSINESS MACHINES, INC. TRS-80 & RADIO SHACK ARE TRADEMARKS OF TANDY CORPORATION.

918/825-4844

AMERICAN SMALL BUSINESS COMPUTERS

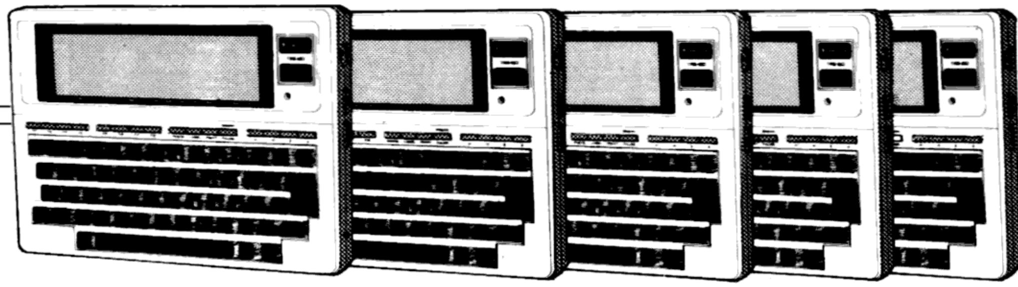
118 SO. MILL ST
PRYOR, OK 74361

NEED INFO QUICK?

Then just call our automated TOLL FREE answering machines. Leave your name, address, and phone number. Requests will be mailed 1st Class the same day. **Only requests for product information will be processed; all others will be disregarded by our answering service.**

CALL **1-800-331-3896**

In Oklahoma, Call
1-918-825-4844 ✓217



Inside the Model 100

by David P. Sumner

To help you write sophisticated programs on your Model 100, I'm providing a fairly extensive memory map, some relevant documentation, and a list of the tokens for the Basic keywords.

One word of caution: It's possible that some versions of the Model 100 have a ROM slightly different from the one in my machine. You should also be aware that some locations have multiple uses, some of which may not be documented here.

Running a Machine-Language Program

You will find the addresses of several routines provided in the memory map useful (see Table 1) primarily for machine-language programs, but it's important to know how to access them from Basic. The secret is in the Basic command Call. To run a machine-language program that resides at location 30306 (the Model 100's sound routine), type in the Basic statement CALL 30306 and you'll hear a beep. Not all Calls are so direct. Often you'll need to pass data to the routine that you're calling. The Model 100 provides nicely for this. A statement of the form

CALL address,X,Y

initiates the execution of the machine code stored at the given address, but first it assigns the value in X to the accumulator, and the value in Y to the register pair HL. Thus, the value of X can only be between zero and 255 (so that it can fit into the 8-bit accumulator), and the value of Y can be any 16-bit integer from zero to 65535.

For example, a Call to location 32 prints the contents of the accumulator (considered an ASCII character) on the display. If you simply type CALL 32, nothing happens since the accumulator contains zero. However, if you type CALL 32,65 you'll see the letter A appear on the screen. You've put 65, the ASCII code for an A, into the accumulator prior to the Call. The effect of CALL 32,X is the same as printing CHR$(X). In fact, since PRINT CHR$(7) produces a beep, CALL 32,7 also results in a beep. Another example: The routine located at 10161 displays the message located at the address pointed to by the register pair HL. If you look at the memory map, you'll see that the 100 stores many messages throughout the ROM. For example, try typing CALL 10161,0,3442. The computer displays the message "extra ignored." Type CALL

The Key Box

The programs in "Blackjack" and "Program Length" will run in 8K RAM.

Table 1. The memory map.

NOTE: Material in single quotes represents ASCII text.
A "+" after a location indicates a 2-byte pointer.

0	(RST 0) JMP 32051
3	'Menu'
8	(RST 1) Test for special character and fall into RST2.
16	(RST2) Set pointer to next character of BASIC text. (JMP 2136)
24	(RST3) Compare registers HL,DE directly
32	(RST 4) PRINT ASCII contents of accumulator.
36	(TRAP) Handles power-down. User interface at 62978.
40	(RST 5) JMP 4201 Checks variable type.
44	(RST 5.5) JMP 62969
48	(RST 6) Returns sign of FAC1. (JMP 13276)
52	(RST 6.5) JMP 28076 (62972 RAM vector interface)
56	(RST 7) Executes routine indicated by next byte. (JMP 32767)
60	(RST 7.5) Updates timer, adjusts power-down values etc. (JMP 6962, user can interface at 62975).
64	BASIC function addresses.
128	BASIC Keywords with high bit set in the first character of each word.
610	BASIC command addresses.
750	Table of BASIC addresses.
796	2-Byte error codes.
858	Initial values for pointers 62960 -63103.
1003	'Error'
1010	'in'
1014	'ok'
1019	'Break'
1094	Syntax error. Other entry points at 1100, 1103 .. 1115 for other errors.
1117	Error message based on contents of E register.
1245	PRINTs error messages (accumulator holds a value from 28 to 58).
1520	Builds BASIC line pointers.
1576	Enter with DE containing a line number. Exit with BC containing

Table 1 continued

10161,0,32676, and the TRS-80 logo will appear on the display.

PEEK, POKE, and Pointers

There is more of interest here than just the addresses of particular routines. Basic uses many locations to store data such as the values of variables, the text of Basic programs, and pointers that determine how the computer behaves. The PEEK command accesses the values in these locations, and the POKE command can change them.

For instance, the location 63368 contains the horizontal position of the cursor. If you let X = PEEK(63368) in a Basic program, the effect is the same as if you had used the expression X = POS(0). Try the following short program as an illustration. (Don't overlook the semicolon in line 20.)

```
10 FOR I = 1TO100
20 PRINT PEEK(63368);
30 NEXT I
40 END
```

If the number stored in location 64173 is zero, then you cannot use the label line. With a POKE command you can put a zero into this location. Try the following experiment. First enter Basic from the main menu and then press the label key. You'll see the key menu displayed on the label line. Now clear the label line by pressing the label key again. Finally, type POKE 64173,0 and press the enter key. Pressing the label key now has no effect. If you type

```
POKE 64173,255
```

things return to normal.

Many of the pointers point to addresses, line numbers, and other 16-bit integers. The Model 100 stores such addresses with the low byte of the integer followed by the high byte. For instance, the computer stores the line number where a break occurred in locations 64426–64427. To determine the line number just type PRINT PEEK(64426) + 256*PEEK(64427).

To determine the starting location of the currently active program, type PRINT PEEK(63100) + 256*PEEK(63101).

Floating Point Operations

In order to use the floating point routines, you must first place the proper values in the floating point accumulators. For instance, to find the square root of 2 using a call to location 12378, you must place 2 in the primary floating point accumulator FAC1 prior to the call. After the call, the value of the square root of 2 resides in FAC1, and you must move it to the proper variable.

Table 1 continued

	the location of the line, and HL the location of the next line. The carry flag is used to indicate if the line exists.
1606	Tokenize BASIC text.
1830	FOR
1899	TO
2112	BASIC command dispatcher; accumulator holds token.
2136	Set pointer to BASIC text.
2162	DEF (general)
2177	DEFDBL
2182	DEFINT
2198	DEFSNG
2207	DEFSTR
2267	FC error.
2283	Used by GOTO, GOSUB, and RUN to determine the 2-byte value of the line (in ASCII) to branch to. The HL register points to the ASCII string of digits, the 16-bit integer is returned in DE.
2319	RUN
2334	GOSUB
2358	GOTO (contains a bug!)
2381	Jump here if UL error.
2406	RETURN
2462	DATA
2464	REM,ELSE
2499	LET
2607	ON (general)
2612	ON ERROR
2736	RESUME
2831	ERROR
2842	IF
2894	LPRINT
2902	PRINT
3073	TAB(
3141	LINE (general)
3152	LINE INPUT
3188	'?Redo from start'
3225	INPUT #
3235	INPUT
3289	READ
3442	'Extra ignored'
3625	>,=,<
3926	ERL
3966	VARPTR
4072	If character in M is lowercase, the upper case equivalent is returned in the accumulator.
4236	OR
4247	AND
4258	XOR
4269	EQV
4277	IMP
4296	LPOS
4302	POS
4305	Store accumulator as low byte of 16-bit integer.
4352	INP
4364	OUT
4398	ASCII text (string/variable) converted to integer.
4411	LLIST
4416	LIST
4514	PRINTs from buffer until 0 byte is encountered.
4522	Put data into buffer until a 0 byte is reached.
4740	PEEK
4747	POKE
4811	Saves registers, waits for

Table 1 continued

Table 1 continued

	character of input, and restores registers.
4848	PASTE.
5029	Toggles the label line.
5138	Break routine.
5145	POWER
5169	Power off /Returns to program on power-up.
5201	Power off
5209	POWER CONT
5225	Sets power-down values.
5288	Preliminary tape I/O routine.
5290	Called at the end of tape I/O.
5296	Returns byte from tape in accumulator.
5313	Sends byte in accumulator to tape.
6281	EOF
6404	TIME$
6436	DATE$
6485	DAY$
6520	ASCII of days stored here.
6553	Converts byte pointed to by DE to an ASCII digit. Result placed in M.
6571	TIME$ (as command/assignment).
6589	DATE$ (as command/assignment).
6641	DAY$ (as command/assignment).
6547	MAX RAM
6776	IPL
6814	COM,MDM
6851	KEY ON/OFF/STOP
6927	ON TIME$
7096	KEY (general)
7101	KEY LIST
7136	PRINTS B PRINTable-characters starting at address in HL.
7182	Defines function key. HL points to string, accumulator holds key number minus one.
7255	PSET
7270	PRESET
7277	Line drawing routine.
7519	Subroutine for PRINT @
7568	CSRLIN
7579	MAX (general).
7602	MAX FILES
7609	HIMEM
7619	WIDTH
7621	SOUND (general)
7653	SOUND OFF
7654	SOUND ON
7660	MOTOR
7667	Turns motor on.
7669	Turns motor off.
7674	CALL
7714	SCREEN
7774	LCOPY
7994	FILES
8081	KILL
8247	NAME
8446	NEW
8832	CSAVE (general)
8856	CSAVEs current BASIC program.
8889	Sends the DE bytes starting at location in HL to tape.
8908	SAVEM
8925	CSAVEM
9079	CLOAD (general)

Table 1 continued

The routines that you need for these actions all appear in the map. You can use the machine code below to find the square root of the variable pointed to by the HL register pair.

```
PUSH H        ; We will need this address later.
CALL 12740    ; Put the variable into FAC1.
CALL 12378    ; Take the square root.
POP H         ; I told you we needed that address.
CALL 12746    ; Replace the value by its square root.
RET           ; That's all.
```

Access the program above from Basic by using the next program, being sure to protect it first by the command CLEAR 256,61999. This prevents Basic from destroying the machine code starting at location 62000.

```
  5 GOSUB 100
 10 INPUT"X ";X
 20 A = VARPTR(X) + 65536
 30 CALL 62000,0,A
 40 PRINT X
 50 END
100 FOR I = 0 TO 11
110 READ V
120 POKE 62000 + I,V
130 NEXT I
140 RETURN
150 DATA 229,205,196,49,205,90,48,225,205,202,49,201
```

This program assumes that you've placed the machine-language routine at location 62000. Line 20 determines the memory location of the variable X, and line 30 passes this address to the register pair HL before executing the routine at 62000.

Lines 100–150 POKE the machine code described earlier into memory.

Now run the program, and you'll see that the value of the variable X is replaced by its square root.

Some of the other floating point functions seem to require the setting of the variable type flag prior to a Call.

Interrupts and the RST Instructions

Like the 8080, the Model 100's 8085 chip contains eight restart instructions that are effectively 1-byte Call instructions. (See Table 1.)

Unlike the 8080, the 8085 has several interrupt routines known by the mnemonics RST 5.5, RST 6.5, RST 7.5, and TRAP. The addresses of each of these resides in the low bytes of ROM. However, the Model 100 allows a user to intercept these routines by placing a Jump instruction in a particular location in RAM. These interface vectors are documented in the map.

The Basic Commands

The memory map contains the addresses of most of the Basic commands and functions. The important thing to realize is that most Basic commands expect the accumulator to contain the next byte of Basic text upon entry to the routine. Some commands, such as Beep and End, aren't modified by additional text. A simple call to the addresses of these routines has the desired effect. On the other hand, the actions of some commands depend upon the text that follows them. For example, consider the Key command. When this statement appears in a Basic program or in immediate mode, several things may

follow it. For instance, KEY ON, KEY OFF, KEY LIST, or KEY 5,"PRINT" are all possible. At the time the program executes the command, the computer has already tokenized the keywords, and the Key routine expects to find the appropriate token or text in the accumulator. The token for ON is 151, and so typing CALL 7096,151 produces the effect of KEY ON. Similarly, since the token for LIST is 165, CALL 7096,165 will result in the same action as if you had entered KEY LIST. Unfortunately, you'll find that not all calls to Basic command locations respond as expected. So be cautious when experimenting with these addresses.

Using the Memory Map

The examples given earlier should help you take advantage of much of this information. You cannot use all of the routines in the ROM directly, however. Many of them, like the floating point operations, require setting up special conditions prior to the call. And, although I would strongly urge you to experiment with your computer, be sure you have saved any important data or programs on tape first. An inappropriate call or POKE can easily garble a program or cause the computer to hang. Of course, you will not damage the computer by such a crash, but you may be unable to restart the computer without using the memory kill button on the bottom of the computer.

About the Map

Notice that items enclosed in single quotes represent actual ASCII text. For instance, the word 'Error' is stored in memory starting at location 1003. The letters H,L,D,E,B,C, and A refer to registers in the 8085. ■

David P. Sumner can be reached at 1009 Walters Lane, Columbia, SC 29209.

Table 1 continued

9235	Gets DE bytes from tape and stores them at the location pointed to by HL.
9302	CLOAD?
9345	'Verify failed'
9361	LOADM, RUNM
9383	CLOADM
9538	Moves B bytes from address in HL to address in DE in an increasingz manner.
9587	CLOADM?
9685	'Top End Exe'
9697	'Exe:'
9982	'Found'
9989	'Skip'
10042	STR$
10161	PRINTs message pointed to by HL. Message ends in quote or 0-byte
10444	String addition.
10508	Moves L bytes from address in BC to address in DE (increasing).
10563	LEN
10575	ASC
10597	CHR$
10605	STRING$
10638	SPACE$

Table 1 continued

✓ *See List of Advertisers on Page 291*

Table 1 continued

10667	LEFT$
10716	RIGHT$
10726	MID$
10759	VAL
10807	INSTR
11084	FRE
11113	FAC1 <-- FAC1-FAC2.
11128	FAC1 <-- FAC1+FAC2.
11519	FAC1 <-- FAC1*FAC2.
11719	FAC1 <-- FAC1/FAC2.
12006	Moves C bytes starting from address in HL to address in DE in a decreasing manner.
12015	COS
12041	SIN
12120	TAN
12145	ATN
12239	LOG
12378	SQR
12452	EXP
12606	RND
12686	MOVE FAC1 to 64633-64640.
12692	Adds memory to FAC1.
12698	Subtracts memory from FAC1.
12704	FAC1 <-- FAC1*FAC1 (squares FAC1)
12707	Multiplies memory and FAC1.
12725	Moves FAC1 to FAC2.
12728	Moves number at address HL to FAC2.
12737	Moves FAC2 to FAC1.
12740	Moves number at address HL to FAC1.
12746	Moves FAC1 to address in HL.
12852	Saves FAC1 on the stack.
12892	Floating point constants.
12942	.25
12974	pi/2
12990	Square root of 3.
13015	1
13203	2*pi.
13298	ABS
13309	Negate FAC1.
13319	SGN
13417	Moves B bytes of memory from address in DE to address in HL in an increasing manner.
13426	Moves B bytes of memory from address in DE to adress in HL in a decreasing manner.
13569	CINT
13610	CSNG
13754	CDBL
13893	FIX
13908	INT
14804	PRINTs the 16-bit integer stored in the register pair HL.
16288	TIME$ ON
16306	TIME$ OFF
16313	TIME$ STOP
16393	Clears COM and TIME$ locations. Clears KEY definitions and sets 63060 to 0

Table 1 continued

Table 1 continued

15758	
16511	RESTORE
16538	STOP
16543	END
16602	CONT
16625	Determines if HL points to an alphabetic character.
16626	Determines if the accumulator contains an alphabetic character.
16633	CLEAR
16756	NEXT
16930	PRINTs CHR$(13)+CHR$(10)
16937	BEEP
16941	PRINTs CHR$(11)
16945	CLS
17001	Turns on reverse video.
17006	Turns off reverse video.
17008	PRINTs CHR$(27)+CHR$ of accumulator.
17034	Clears the label line.
17064	PRINTs label line.
17994	Puts input data into input buffer.
18315	DIM
18320	Get variable name and type.
18833	USING
19268	PRINTs ASCII character in accumulator.
19322	Sends byte in accumulator to the printer.
19360	Sends carriage return to printer
19434	INKEY$
19659	OPEN
19824	LOAD
19825	MERGE
19919	SAVE
20008	CLOSE
20110	INPUT$
20591	LFILES
20725	'CRT CAS COM WAND LPT MDM RAM'
20806	TELCOM program.
20558	Enter here or one of 20561, 20563...20579 in case of file errors.
20593	DSKO$
20595	DSKI$
20860	'Telcom'
20869	'STAT'
20874	'TERM CALL RFIND MENU'
20900	'Find Call Stat Term Menu'
20906	'x pps'
21060	'Calling'
21890	'Full Half Echo'
21909	'Wait'
22353	'File to upload File to download
22385	'aborted'
22396	'No file'
22406	'Disconnect'
22417	PRINTs message. Sends a cr if not at start of a line.

Table 1 continued

↗276

↗ See List of Advertisers on Page 291

Table 1 continued

22423	MENU
22574	PRINTs number of free bytes.
22580	PRINTs 'Select:' and responds appropriately to input.
23128	PRINTs starting at address in HL until a zero byte is reached.
23164	Sets up function keys.
23273	Months stored in ASCII.
23312	'Microsoft'
23332	'Select:'
23366	Original key definitions stored here. FILES LOAD" SAVE " RUN LIST MENU
23400	ADDRSS (program)
23407	SCHEDL (program)
23450	PRINTs 'Not found press space bar for menu"
23758	ADDRS.DO
23767	'Not found'
23852	'Call'
23920	PRINTs date and time at top of screen and updates it until a key is pressed.
24046	TEXT (program)
24051	Requests a file to edit.
24085	'File to edit'
24106	'FIND LOAD SAVE COPY CUT SEL MENU'
24145	EDIT (program)
24367	Waits for a space keypress and then returns.
24376	'Text ill-formed'
24395	'Press space bar for TEXT'
24753	'Memory full
26051	Moves memory starting at address in HL to address in DE until a 0 byte is reached.
26062	'No Match'
26071	'String'
26380	'Width'
26421	'Save to:'
26579	'Load from:'
27611	Moves BC bytes from address in HL to address in DE increasing.
27636	'BASIC TEXT TELCOM ADDRSS SCHEDL Suzuki Hayashi'
27721	BASIC (as called from MENU)
27795	Temporarily saves the function keys.
27804	Restores the function keys.
29156	Adds a character to the keyboard buffer.
29250	Returns ASCII of keypress in the accumulator; does not wait-returns 0 if no keypress.
29381	SOUND routine. DE contains pitch, and B contains duration.
29772	Turns on the pixel (x,y) where D contains x and E contains y.
29773	Turns off the pixel (x,y) where D contains x and E contains y.
30326	Toggles the speaker.
30306	Equivalent to BEEP.
30481	Character set 5-bytes each.
31729	Keyboard matrix
32051	Initialization
32231	Initialize pointers (cold start)
32422	PRINTs TRS-80 logo.
32428	PRINTs number of free bytes.
32523	MAX FILES=
32664	' bytes free'
32676	TRS-80 logo stored in ASCII.
40960-62959	User RAM in 24K machine.
62964+	HIMEM value.
62966	Code called at 32145 and 32197.
62969	RAM vector for RST 5.5.
62972	RAM vector for RST 6.5.
62975	RAM vector for RST 7.5.
62978	JMP 5169
62981	Code called at 32108 and 32292.
62991	Code called at 896.
63012	Code
63024	8-bytes KEY ON/OFF flags; ON=1

Table 1 continued

Table 1 continued

	OFF=0
63033	Vertical Print position 0 to 7.
63034	Next horizontal print position 0-39.
63035	Number of active lines 0-8.
63036	Number of active columns 0-40.
63037	Label line flag; 0=unused, 255=used.
63038	Inhibits return to first line if not zero.
63040	Cursor line.
63041	Horizontal print position.
63048	Reverse video if not zero.
63054	x-pixel set.
63055	y-pixel set.
63063	Power-down value (constant).
63070	Code (to 63078)
63079	OUT/INP self-modifying code.
63090	Error code
63092	Value of LPOS
63093	Output flag; 0=display, 1=printer.
63096+	Top of available RAM.
63098+	Current BASIC line number; 65535 stored here if no program is running.
63100+	Start ot current BASIC program text.
63104	End of statement marker (: OR 0)
63105	Multi-purpose buffer area. Tokenized text starts at 63105. Input buffer starts at 63109 and extends to 63362.
63368	Value of POS
63369	Function key definitions currently active. (extends to 63497)
63498	Function key defintions used by BASIC. (extends to 63626)
63628+	Pointer to start of PASTE text.
63639	Start of work area.
63785	Day of the month low digit here, high digit in 63786.
63787	Current day of the week (e.g. 3=wed.)
63788	Current month - decimal 1 to 12.
63789	Current year stored with low decimal value here and the high decimal value in 63790
63791	Timer; decreases from 125 to 0.
63792	Timer decreases from 12 to 0.
63793	Power-down countdown value (varies).
63795	The computer stores the current time starting here with the low digit of the number of seconds. A numerical value - not ASCII.
63796	The high digit of number of seconds.
63797	Low digit of number of minutes.
63798	Time and date continues here with one decimal value per location.
63805	6-byte value of TIME$ for ON TIME$ statement (in reverse order).
63812	COM ON/OFF flag.
63813+	Address of COM ON routine.
63815	TIME$ ON/OFF flag.
63816+	Address of TIME$ ON routine.
63818	ON/OFF flag and routine address

Table 1 continued

✓ See List of Advertisers on Page 291

Table 1 continued

```
        of function keys; 3 bytes per
        key. (extends to 63841)
63842   Files in the format: address, 6-
        character name, 2-byte extension

63898+  Address of BASIC program that has
        not been saved to RAM. (Suzuki)
63909+  Address of Hayashi, points to the
        end of documents.
64173   Label line enable flag; enabled
        if not zero.
64175   Name of IPL program.
64190+  Used for temporary storage of
        stack pointer.
64208+  Length of CLOADed/CSAVEd program
64357   BASIC variable type.
64404+  Line number of active DATA
        statement.
64409+  Location of BASIC variable for
        assignment statement.
64411+  Start of current BASIC statement
64413+  2 less than value in 63096-63097
64415+  Line where error occured.
64417+  Most recently entered/listed line

64419+  Location of statement where error
        occured.
64421+  Location of ON ERROR GOTO line.
64423   Error status flag.
64424+  End of BASIC expression.
64426+  Line where Break occured.
64428+  Location after error (contains 0
        or 58).
64430+  Start of documents.
64434+  Start of variables.

        The region from 64536 to 64640 is
used for floating point computations.

64536   FAC1 (floating point accumulator)
        8-bytes.
64617   FAC2 (floating point accumulator)
        8-bytes.
64642   Maxfiles.
64659   Name of current BASIC program; 6-
        bytes.
64668   Name of program loaded from tape;
        6-bytes.
64904   Start of date and time stored in
        ASCII. Used for Menu display.
        Not used for TIME$.
65024   Start of screen memory.
65348   Sound flag; on=0, off=175.
65349   Cassette on/off flag.
65424   Holds the value 2 as long as a
        noncontrol key is held down.
65429   Devoted to the number keys. Also
        uses location 65430
65431   Bits are set here according to
        which of the following keys are
        pressed: SPACE,DEL,TAB,ESC,
        PASTE,LABEL,PRINT,ENTER.
65432   Pressing a function key sets the
        corresponding bit in this
        location.
65441   Behaves like 65432.
65442   The following keys set bits in
        this location: SHIFT,CTRL,GRPH,
        CODE,NUM,CAPS LOCK.
65446   Code (not ASCII) for most
        recently pressed key.
65450   Number of characters in keyboard
        buffer.
65451   Keyboard buffer (32 byte maximum)
        Odd bytes contain ASCII values.
        A 255 in an even byte indicates
        a function key.
65515   Used to store 5-byte character
        code.
```

Token	Keyword	Token	Keyword	Token	Keyword
128	END	171	DATE$	213	AND
129	FOR	172	DAY$	214	OR
130	NEXT	173	COM	215	XOR
131	DATA	174	MDM	216	EQV
132	INPUT	175	KEY	217	IMP
133	DIM	176	CLS	218	MOD
134	READ	177	BEEP	219	\
135	LET	178	SOUND	220	>
136	GOTO	179	LCOPY	221	=
137	RUN	180	PSET	222	<
138	IF	181	PRESET	223	SGN
139	RESTORE	182	MOTOR	224	INT
140	GOSUB	183	MAX	225	ABS
141	RETURN	184	POWER	226	FRE
142	REM	185	CALL	227	INP
143	STOP	186	MENU	228	LPOS
144	WIDTH	187	IPL	229	POS
145	ELSE *	188	NAME	230	SQR
146	LINE	189	KILL	231	RND
147	EDIT	190	SCREEN	232	LOG
148	ERROR	191	NEW	233	EXP
149	RESUME	192	TAB(	234	COS
150	OUT	193	TO	235	SIN
151	ON	194	USING	236	TAN
152	DSKO$	195	VARPTR	237	ATN
153	OPEN	196	ERL	238	PEEK
154	CLOSE	197	ERR	239	EOF
155	LOAD	198	STRING$	240	LOC
156	MERGE	199	INSTR	241	LOF
157	FILES	200	DSKI$	242	CINT
158	SAVE	201	INKEY$	243	CSNG
159	LFILES	202	CSRLIN	244	CDBL
160	LPRINT	203	OFF	245	FIX
161	DEF	204	HIMEM	246	LEN
162	POKE	205	THEN	247	STR$
163	PRINT	206	NOT	248	VAL
164	CONT	207	STEP	249	ASC
165	LIST	208	+	250	CHR$
166	LLIST	209	−	251	SPACE$
167	CLEAR	210	*	252	LEFT$
168	CLOAD	211	/	253	RIGHT$
169	CSAVE	212	Λ	254	MID$
170	TIME$				

** Basic reads ELSE as :LSE, with LSE = 145.*

Table 2. Basic keywords and tokens.

See List of Advertisers on Page 291

Make Your Own Modem Cable

by Carl Oppedahl

While the Radio Shack modem cable for the Model 100 comes with one free hour each of CompuServe and Dow Jones services, this may be of little use to you if you already subscribe, or if you need more than one modem cable. For these reasons I built my own.

The circuit diagram for the Radio Shack cable appears in Table 3. One end is an exotic 8-pin DIN plug (see Photo 1) whose spacings are slightly different from those of the 8-pin cassette DIN plug. Look closely at the top panel of the Model 100: The difference lies in the placement of pins 6, 7, and 8. In the phone jack pins 7 and 8 sit directly below 1 and 3, while in the cassette jack pins 6, 7, and 8 are somewhat closer together.

Extending from the 8-pin plug are two conventional-looking modular telephone line cords. You would connect the beige cord to the local phone company dial tone, as with a wall jack, and the silver cord to a conventional telephone if you were using the computer as an automatic dialer for voice calls.

Unplug the modem cable from the Model 100, and the telephone connected to the silver cord will go dead. So Radio Shack provides a shorting connector to mate with the 8-pin plug in place of the Model 100. It connects pins 1 and 7, temporarily making the modem cable into a rather expensive telephone extension cord.

But with a bit of drilling and soldering, you can make your own cable at a cost of only a few dollars. Even if you buy all the parts new from Radio Shack and want to duplicate all the functions, including autodialing for voice telephone calls, you'll spend only $10.88. (See Table 4.)

Since you need only pins 1, 3, and 7 (and you can omit pin 1 if you don't need the silver cord) you can pry loose one of pins 2, 4, or 5 to be used in position 7. However, it's difficult to remove pins from the black plastic. I ended up buying an extra 5-pin DIN plug, and simply cracked apart the plastic to get a spare pin to mount into a newly-drilled hole in the other.

After you extract an extra pin from a DIN plug, drill a hole in the 5-pin plug so that you can insert the new pin and glue it in place. This requires a 5/64-inch drill bit and a steady hand. Before drilling, slip off the plastic sleeve of the DIN plug by lifting the tab above pin 2. Then separate the two halves of the metal barrel inside. This exposes the black plastic carrier containing the five pins, which requires a hole for the new pin 7. The important thing is to drill the hole directly below pin 1, so that the new pin 7 fits into the matching hole in the Model 100 phone jack.

Then, mix up some epoxy glue. (I used a brand that sets in 10 minutes with satisfactory results.) Grasp the extra pin with a tweezer or needlenose pliers, apply glue to it with a toothpick, and insert it carefully into the hole. (See Photo 2.) Hold it parallel to the other pins until the glue has hardened somewhat. Then let it set for the period recommended in the glue instructions.

Next take the modular phone cord and cut it in half. Taking one half, carefully remove about an inch of the outer jacket. Inside you'll find green, red, and probably black and yellow wires. Clip the yellow and black wires (if any) short and strip the red and green wires, which carry what the telephone com-

DIN pin	Modular plug
1	grey cable, green wire
2	NC
3	grey cable, red wire, and beige cable, red wire
4	NC
5	NC
6	NC
7	beige cable, green wire
8	NC

Table 3. Connections in Radio Shack Model 100 modem cable.

Part Description	Current Price
Two 5-pin DIN plugs, part number 274–003	$1.49
Telephone line cord, part number 279–374	$4.95
Inline coupler, part number 279–358	$2.95

Table 4. Parts list for homemade modem cable.

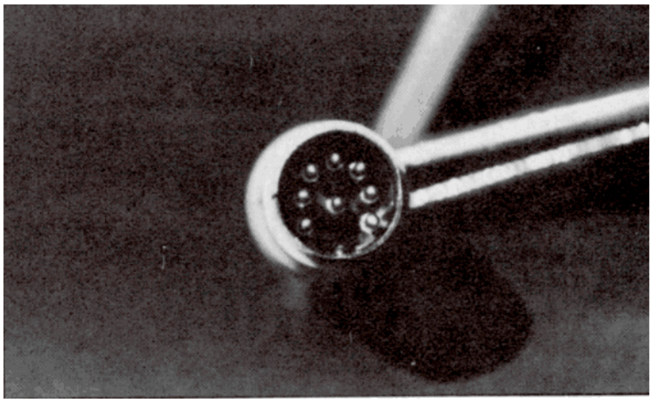


Photo 1. Radio Shack 8-pin DIN plug.

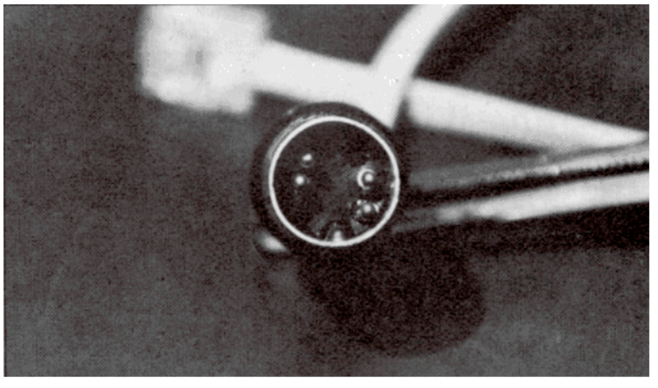


Photo 2. 5-pin DIN plug with new pin 7.

✓ See List of Advertisers on Page 291

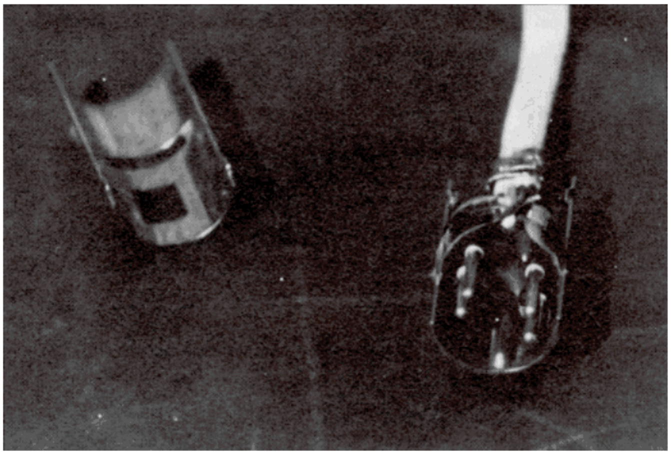


Photo 3. The red wire goes to pin 3, while the green wire goes to pin 7.

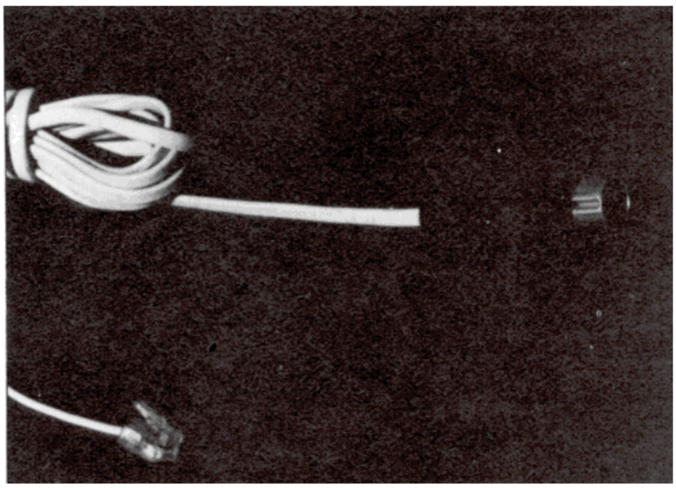


Photo 4. Completed modem cable.

pany calls "ring" and "tip" signals, respectively.

Thread the DIN plastic sleeve onto the phone cord so that later you can slip it onto the DIN plug. Then solder the cord to the plug, connecting the red wire to pin 3 and the green wire to pin 7. Reassemble the metal barrel. (See Photo 3.) Slide the sleeve back on, and the cable should be ready to test. (See Photo 4.)

Testing the cable is easy. If you have access to an ohmmeter, use it to check that none of the DIN pins is shorted to any other, and that none of the modular plug pins is shorted to any other. Then check for continuity along the modem cable—the red wire at the modular plug should connect with pin 3 at the DIN plug, while the green wire should go to pin 7. Then plug the modem cable into the computer and the telephone line.

Call your local Radio Shack store to get the telephone number for Tymnet or Telenet. With the computer in TELCOM, dial that number on an extension phone, and when you hear the high-pitched carrier tone, push F4. Words and letters should appear on the screen. Then push F8, the Bye key, and type Y. The tone should stop. ■

Carl Oppedahl can be reached at 99 Park Ave., New York, NY 10016.

Program Length

by Ronald F. Balonis

Length.BA, an 8K RAM utility program, provides a method to compute program length in order to manage memory space on the Model 100. (See Program Listing 1.)

All memory references are in decimal notation. The user menu directory is located in RAM memory from 63930 to 64139, with each entry using 11 bytes. The first byte of each entry denotes the type of file: 0 is for a killed/empty file, 128 is for Basic program (.BA) files, and 192 for text (.DO) files. The next 2 bytes are the start address of the file in RAM memory, and the remaining 8 bytes are the file name and extension. Disassembly of the file area indicated that 26 is the end of file (EOF) indicator of a text file, and the standard string of three zeros is EOF for a .BA file.

The program conducts a sequential search of the directory for a file match, and then a sequential search of the file for the EOF byte, using PEEKs to read the memory. The menu directory search, while slow, proved acceptable, but the EOF search of the file had a tiresome wait with long files. This was an application in need of some machine language, and so I made my first attempt at a machine-language subroutine on the Model 100.

Borrowing on early Model I techniques, I fashioned a simple machine-language sequential search routine for the EOF search. The program loads it into the Model 100 with data statements, and the Basic PEEK and POKE statements pass the variables to and from the routine. A word or two of warning about my experience with machine language on the Model 100 is in order: Errors, at least of the types I made, cause the computer to do a cold restart, erasing all of the user programs.

Program Operation

Line 15 clears string space and reserves 50 bytes (from MAXRAM to HIMEM 63960 to 63910) for the machine-language subroutine. Lines 30–50 POKE the machine-language program (see Program Listing 2) into this memory space.

Lines 100–180 prompt for a file name, test that it has a valid formation, then construct a menu directory match string of it. The program stores the file names in the menu directory without the period before the extension, with the name left-justified and the extension right-justified, and with spaces filling the middle if necessary. Lines 200–260 do the sequential search of the menu directory using the PEEK function to create a match string. Line 205 tests for a killed/empty file, and line 225 tests for a match of file names. If a match is found, then lines 230–235 get its address. Line 310 passes this address to the machine-language program for the EOF search called in line 320 or 330. Lines 400–430 compute and display the results.

The utility program is simple to use. Just enter a valid file name and in two to three seconds the program displays its length for logging. Press the space bar for another program or to exit. ■

Write to Ronald F. Balonis at 118 Rice St., Trucksville, PA 18708.

```
5 'LENGTH.BA  FIND LENGTH OF A PROGRAM
10 'BY R.F.BALONIS  JULY 16, 1983
15 CLEAR 100,MAXRAM-50:'JULY 17, 1983
20 TITLE$="LENGTH OF A PROGRAM"
25 '-MACHINE LANGAUGE PROG. END SEARCH-
30 FOR I=62911 TO 62958
40  READ Z:POKEI,Z:B$=B$+" "
50 NEXT I
60 DATA 0,0,229,42,191,245,43,
35,126,254,26,194,198
70 DATA 245,195,234,245,229,42,
191,245,43,35,126,254
80 DATA 0,194,213,245,35,126,254,
0,194,213,245,35
90 DATA 126,254,0,194,213,245,
34,191,245,225,201
95 '
100 B$="":FNM$="":ADRS=0
110 CLS:PRINT@10,TITLE$
120 PRINT@82,"ENTER NAME AS IT APPEARS
ON THE MENU"
130 PRINT@165,"<ENTER> TO EXIT   ";
135 INPUT FNM$:IF FNM$="" THEN MENU
140 I=INSTR(FNM$,"."):TYPE$="":TYPE=0
145  IF I=0 THEN 100
150   TYPE$=MID$(FNM$,I+1,2)
155    IF TYPE$="DO" THEN TYPE=1
160     IF TYPE$="BA" THEN TYPE=2
165      IF TYPE=1 OR TYPE=2 THEN 170
ELSE 100
170 FILN$=LEFT$(FNM$,I-1)
175  IF LEN(FILN$)<6 THEN FILN$=FILN$+"
":GOTO 175
180 FILN$=FILN$+TYPE$
185 PRINT@285,"**** SEARCHING DIRECTORY
****";
190 '
195 '--SEQUENTIAL DIRECTORY SEARCH--
200 FOR I=63930 TO 64139 STEP 11
205 B$="":IF PEEK(I)=0 THEN 240
210  FOR II=3 TO 10
215   B$=B$+CHR$(PEEK(I+II))
220  NEXT II
225  IF INSTR(B$,FILN$)<1 THEN 240
230   IL=PEEK(I+1):IH=PEEK(I+2)
235   ADRS=IH*256+IL:I=64139
240 NEXT I
245 IF ADRS=0 THEN 250 ELSE 290
250 PRINT@285,"  **** NOT IN DIRECTORY *
***  ";
260 FOR I=1 TO 1000:NEXT I:GOTO 100
280 '
290 '---FIND THE LENGTH---
300 PRINT@285,"  **** COMPUTING LENGTH *
***  ";
310 POKE62911,IL:POKE62912,IH
320 IF TYPE=1 THEN CALL 62913
330  IF TYPE=2 THEN CALL 62928
340 '
400 CLS:PRINT@10,TITLE$
405 IADR=PEEK(62912)*256+PEEK(62911)
410 PRINT@163,"< "FNM$" > IS "IADR-ADRS"
BYTES LONG"
420 PRINT@287,"**** PRESS <SPACEBAR> ***
*";
430 IF INKEY$=" " THEN 100 ELSE 430
```

Program Listing 1. Length.BA—utility.

	Memory Address	Source Statement		Object Code
L Addr	62911			0
H Addr	62912			0
DO Entry	62913	Push	HL	229
	62914	LD HL,	62911	42,191,245
	62917	Dec	HL	43
	62918	Inc	HL	35
	62919	LD A,	(HL)	126
	62920	Cp	26	254,26
	62922	JNZ	62918	194,198,245
	62925	JP	62954	195,234,245
.BA Entry	62928	Push	HL	229
	62929	LD HL,	62911	42,191,245
	62932	Dec	HL	43
	62933	Inc	HL	35
	62934	LD A,	(HL)	126
	62935	Cp	0	254,0
	62937	JNZ	62933	194,213,245
	62940	Inc	HL	35
	62941	LD A,	(HL)	126
	62942	Cp	0	254,0
	62944	JNZ	62933	194,213,245
	62947	Inc	HL	35
	62948	LD A,	(HL)	126
	62949	Cp	0	254,0
	62951	JNZ	62933	194,213,245
	62954	LD 62911,	HL	34,191,245
	62957	Pop	HL	225
	62958	Return		201

Program Listing 2. Length.BA—machine language.

Blackjack

by Paul Serotta

Blackjack isn't a mere conversion: This program uses the Model 100's unique features—graphics, sound, and interrupt-controlled function keys—to simulate the popular casino card game also known as 21. Like the Vegas version, the game program pits a single player against the house's dealer (the Model 100). The object of the game is to accumulate a hand worth 21 points, or as close to 21 as possible, without going over. The house wins if you draw over 21, or if the dealer's hand is closer to 21 than yours is.

The 8K program starts by asking you to type in your name and press the enter key. From now on, it addresses you personally. The program gives you a stake of $1,000 and asks you to enter your bet. Should you decide to wager the entire amount on the first hand, the buzzer sounds and the program applauds your bravado with an encouraging "Go for it!"

Conversely, a low bet (under $100) merits the program's disdainful "You are cheap!"

After you've entered your bet (in full-dollar amounts only), four boxes appear on the screen. The two upper boxes represent your first two cards; a typical deal might be a king of clubs and a six of hearts. In the lower right-hand box the program displays the dealer's first card, for example, a six of spades. You then have the following options, selected with the 100's function keys: hit, double, stay, and quit. Press the F1 key if you want another card, the F2 key if you want to be dealt another card and double your original bet, the F3 key to play a two-card hand against the dealer, or the F4 key to stop the game altogether.

When you stay, you're electing to play your current hand against whatever the house turns up for itself. The program then keeps dealing itself more cards until it has reached 21—an automatic win, beaten your hand, or gone over. If you lose, the program tells you "You are busted." If you win, it admits "I'm busted—you win!" Should the dealer's hand match your own, the program declares the deal a draw or "push" and no one wins or loses.

The program keeps a running tally of your stakes; after each deal it reminds you of how much money you have available and asks you to enter another bet.

It's impossible to cheat at this Blackjack, by the way. Try wagering more money than you have in your purse or doubling at the wrong time, and the program calls you on it.

When you decide to bail out, press the F4 key and the program totals your winnings—or your losses.

There's one born every minute.■

Contact Paul Serotta at 131 Penrose Drive, Pittsburgh, PA 15208.

Program Listing 3. Blackjack.

```
10 REM ***********************************

20 REM              BLACKJACK
30 REM
40 REM PAUL SEROTTA
45 REM 131 PENROSE DR.
50 REM PITTSBURGH, PA 15208
55 REM
60 REM ***********************************

80 REM
99 REM DIMENSION CARD ARRAY, SET
PLAYER'S AMOUNT OF MONEY
100 DIMC(52):PM=1000
105 REM  CLEAR FUNCTION KEYS (F1-F4)
110 FORLL=63369TO63432:POKELL,0:NEXTLL
112 REM TITLE PAGE
115 CLS:LINE(70,24)-(115,52),1,B:LINE(
120,24)-(165,52),1,B:PRINT@173,"ACE
";:PRINT@181,"JACK ";
125 GOSUB9000:CLS: PRINT@121,"PLEASE
TYPE YOUR NAME AND PRESS 'ENTER'"
140 PRINT:LINE INPUTN$:GOSUB9000
145 KEY OFF: IFPM<=0THENGOTO3000
147 REM INITIALIZE LOGIC VARIABLES & ASK
FOR BET
150 PP=1:CS=2:PA=0:CA=0: PT=0:CT=0:CR=2:
PC=81:CP=201: GOSUB9020:PRINT:PRINTN$;",
YOU HAVE $";PM
160 PRINT@200,"";:INPUT"PLEASE ENTER
YOUR BET";BET:BET=INT(BET)
170 IF BET >PM THEN GOSUB9050:GOTO160
180 IFSGN(BET)=-1ORSGN(BET)
=0THENGOSUB9050:GOTO160
190 IFBET=PMTHENPRINT@293,CHR$(27);"p";"
GO FOR IT !! ";CHR$(27)
;"q";:SOUND4000,15:SOUND3000,10:SOUND200
0,8:FORDL=1TO500:NEXTDL
195 IFBET<100ANDPM>500THENPRINT@291,CHR$
(27);"p";" YOU ARE CHEAP !! ";CHR$(27)
;"q";:SOUND4000,30:SOUND12000,30:FORDL=1
TO500:NEXTDL
200
GOSUB9020:GOSUB9100:GOSUB9110:PP=1:GOSUB
9500:ONKEYGOSUB1000,4000,2000,3000
202 PRINT@100,CHR$(155);"
YOU";:PRINT@220,CHR$(155);"
DEALER";:FORDL=1TO300:NEXTDL:PRINT@100,S
PACE$(6);:PRINT@220,SPACE$(9);
210 PRINT@PC,C$;CHR$(156+SU);:PC=PC+
8:PT=PT+CV: GOSUB9500:PT=PT+
CV:PRINT@PC,C$;CHR$(156+SU);:PC=PC+8
215 IFPA=2THENPA=1:PT=PT-10
220 PP=0:GOSUB9500:HC$=C$+CHR$(156+SU)
:CT=CT+CV:CP=CP+8:GOSUB9500:CT=CT+
CV:PRINT@CP,C$;CHR$(156+SU);:CP=CP-8
225 IFCA=2THENCA=1:CT=CT-10
230 IFPT=21ANDCT<>
21THENPRINT@CP,HC$;:PRINT@280,"BLACKJACK
!$!$!$!$!$!$!              ";:PM=PM+BET+INT
(BET/2)
:FORDL=1TO5:SOUND4000,10:SOUND8000,10:NE
XTDL:GOTO145
240 IFCT=21ANDPT<>
21THENPRINT@CP,HC$;:PRINT@280,"I HAVE
BLACKJACK $$!! - YOU LOSE  ";:PM=PM-
BET:SOUND14000,15:SOUND16000,20:FORDL=1T
O1000:NEXTDL:GOTO145
250
IFCT=21ANDPT=21THENPRINT@CP,HC$;:GOTO221
0
260 KEY ON
270 PRINT@280,"HIT  DBL  STAY
QUIT";:FORWW=1TO200:NEXTWW:PRINT@280,SPA
CE$(39);:FORWW=1TO200:NEXTWW
280 IFNH=1THENNH=0:GOTO145:ELSEGOTO270
999 REM PLAYER HITS
1000 CR=CR+1
1010 IFCR=3THENLINE(94,9)-(142,30)
,1,B:GOTO1050
1020 IFCR=4THENLINE(141,9)-(189,30)
,1,B:GOTO1050
1030 IFCR=5THENLINE(188,9)-(236,30),1,B
1050 PP=1: GOSUB9500:PRINT@PC,C$;CHR$(
156+SU):PC=PC+8:PT=PT+CV
1060 IFPT>21ANDPA<=0THENPRINT@280,N$;",
YOU ARE BUSTED
";:FORDL=1TO10:SOUND12000,DL:NEXTDL:PM=P
M-BET:FORDL=1TO500:NEXTDL:NH=1:RETURN
1070 IFPT>21ANDPA>0THENPA=PA-1:PT=PT-10
1100 IFCR<>5THEN RETURN
1200 PRINT@280,"5 CARD CHARLIE - YOU WIN
!!";:FORDL=1TO8:SOUND5000,DL:SOUND15000,
DL:NEXTDL:PM=PM+BET+INT(BET/2)
```

Listing 3 continued

Listing 3 continued

```
:NH=1:RETURN
1999 REM COMPUTER'S LOGIC
2000 KEY OFF
2005 PRINT@280,SPACE$(38);
2010 PP=0:PRINT@CP,HC$:CP=CP+16
2020 WC=94: IFCT>16THEN2200
2050 CS=CS+1:
IFCS=6THENGOTO2206:ELSELINE(WC,32)-(WC+
48,53),1,B:GOSUB9500
2060 WC=WC+47:CT=CT+CV:PRINT@CP,C$;CHR$(
156+SU);:CP=CP+8
2070 IFCT>21 AND CA>0 THENCA=CA-1:CT=CT-
10
2075 IFCS=6ANDCT<22THENGOTO2206
2080 IFCT<=16THEN2050
2199 REM WHAT HAPPENED?
2200 IFCT>21THENPRINT@280,"I'M BUSTED --
- YOU WIN
!!";:SOUND14000,20:SOUND10000,20:SOUND50
00,20:SOUND3500,20:PM=PM+BET:GOTO2500
2206 IFCS>=5THENPRINT@280,"I'VE GOT A 5
CARD CHARLIE  -- YOU LOSE";:PM=PM-
BET:SOUND7500,20:SOUND5000,15:SOUND10000
,20:GOTO2500
2210
IFCT=PTTHENPRINT@280,"PUSH............
";:FORDL=7000TO10000STEP1000:SOUNDDL,8:N
EXTDL:GOTO2500
2220 IFCT>PTTHENPRINT@280,"I WIN
$$$$$$$$";:SOUND16000,25:PM=PM-
BET:GOTO2500
2230 IFCT<PTTHENPRINT@280,"YOU WON
!!!!!";:SOUND3000,20:SOUND8000,20:SOUND3
000,20:PM=PM+BET
2500 FORDL=1TO1250:NEXTDL: IFCO>
25THENGOSUB9100
2505 NH=1: RETURN
2999 REM QUIT
3000 CLS:PRINT@80,"BYE ";N$:IFPM>
=1000THENPRINT@200,"YOU WON $";PM-
1000:END:ELSE
3010 PRINT@200,"YOU LOST $";ABS(PM-1000)
:END
3999 REM PLAYER DOUBLES
4000 IFCR>2THENPRINT@280,"YOU CAN'T
DOUBLE NOW --- DUMMY
!!";:SOUND7500,20:SOUND13000,20:FORDL=1T
O1000:NEXTDL:PRINT@280,SPACE$(38)
;:RETURN
4002 IFBET*2>PMTHENPRINT@280,"YOU DON'T
HAVE ENOUGH MONEY ****
";:SOUND3000,10:SOUND15000,10:SOUND8000,
10:FORDL=1TO1000:NEXTDL:PRINT@280,SPACE$
(35);:RETURN
```

Listing 3 continued

Listing 3 continued

```
4005 PP=1:CR=CR+1:LINE(94,9)-(142,30)
,1,B:GOSUB9500
4010 PRINT@PC,C$;CHR$(156+SU):PT=PT+
CV:BET=BET*2
4020 IFPT>21ANDPA<=0THENGOTO1060
4030 IFPT>21ANDPA>0THENPT=PT-10
4050 GOTO2000
8999 REM PRINT WELCOME
9000 FORLL=1TO5: PRINT@50,"WELCOME TO
";CHR$(27);"p";"BLACKJACK";CHR$(27);"q"
9010 FORDL=1TO60:NEXTDL:BEEP:
PRINT@50,SPACE$(20)
;:FORDL=1TO60:NEXTDL,LL:RETURN
9015 REM PRINT NAME OF GAME ON FIRST
LINE
9020 CLS: LINE(0,0)-(239,7)
,1,BF:PRINT@15," BLACKJACK ":LINE(0,7)-(
239,7):RETURN
9030 REM ANY CHEATING ??????
9050 PRINT@280,"NO CHEATING IN THIS GAME
!";:SOUND4000,10:SOUND10000,10:SOUND1600
0,10
9060
FORDL=1TO750:NEXTDL:PRINT@223,SPACE$(96)
;:RETURN
9099 REM SHUFFLE THE CARDS
9100 CO=0: FORLL=1TO52:C(LL)
=1:NEXTLL:RETURN
9105 REM DRAW THE INITIAL 4 CARDS
9110 LINE(0,9)-(48,30),1,B:LINE(47,9)-(
95,30),1,B:LINE(0,32)-(48,53),1,B:LINE(
47,32)-(95,53),1,B
9115 RETURN
9499 REM PICK A CARD
9500 SEC=VAL(RIGHT$(TIME$,2))
:FORI=1TOSEC:DUM=RND(1):NEXTI:RN=INT(RND
(1)*53)
9510 IFC(RN)=0THEN9500
9520 CO=CO+1:C(RN)=0:SU=RNMOD4:
9530 IFRN>4THENGOTO9535:ELSEC$="ACE
":CV=11
9532 IFPP=1THENPA=PA+1:ELSECA=CA+1
9533 GOTO9600
9535 IFRN>40THENGOTO9540:ELSEC$=STR$(INT
(RN/4))+" ":CV=INT(RN/4)
9537 IFRNMOD4<>0THENC$=STR$(INT(RN/4)+1)
+" ":CV=INT(RN/4)+1
9538 GOTO9600
9540 IFRN<45THENC$="JACK
":CV=10:GOTO9600
9545 IFRN<
49THENC$="QUEEN":CV=10:GOTO9600
9550 C$="KING ":CV=10
9600 RETURN
```

STATEMENT OF OWNERSHIP, MANAGEMENT AND CIRCULATION

1. Title of publication, 80 Micro. A. ISSN 07447868. 2. Date of filing, Oct. 1, 1983. 3. Frequency of Issue, Monthly. A. No. of issues published annually, 12. B. Annual subscription price, $36.00. 4. Location of known office of publication, 80 Pine Street, Peterborough, Hillsboro County, NH 03458. 5. Location of headquarters or general business offices of the publishers, Same. 6. Names and complete addresses of publisher, editor, and managing editor. Publisher and Editor, Wayne Green, 80 Pine Street, Peterborough, NH 03458. Managing Editor, Eric Maloney, 22A Pine Street, Peterborough, NH 03458. 7. Owner, International Data Group, PO Box 1450, 5 Speen Street, Framingham, MA 01701; Patrick J. McGovern, PO Box 1450, 5 Speen Street, Framingham, MA 01701. Stockholders, Wayne Green, Peterborough, NH 03458. 8. Known bondholders, mortgagees, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages or other securities. None. 9. For completion by nonprofit organizations authorized to mail at special rates (Section 411.3 DMM). The purpose, function and nonprofit status of this organization and the exempt status for Federal income tax purposes (Check one) Not applicable. 10. Extent and nature of circulation. (X) Average No. copies each issue during preceding 12 months. (Y) Actual No. copies of single issue published nearest to filing date. A. Total No. of copies printed (X) 148,667 (Y) 164,029 B. Paid circulation 1. Sales through dealers and carriers, street vendors and counter sales (X) 48,160 (Y) 73,415 2. Mail subscription (X) 76,634 (Y) 76,179 C. Total paid circulation (X) 124,794 (Y) 149,594 D. Free distribution by mail, carrier or other means, samples, complimentary, and other free copies (X) 807 (Y) 837 E. Total distribution (X) 125,601 (Y) 150,431 F. Copies not distributed 1. Office use, left over, unaccounted, spoiled after printing (X) 7,980 (Y) 13,598 2. Returns from news agents (X) 15,086 (Y) 0 G. Total (X) 148,667 (Y) 164,029.

Monitor 100 Changes

I found the "Monitor 100" program (August 1983, p. 178) particularly interesting, but to make it work I had to make two changes to the printed version.

First, statement 58 is redundant and should be deleted. Also, in line 200, MID$(0$,3,4) should be MID$(0$,2,4).

The Model 100 is an excellent complement to my big computer (a Kaypro II set up as an RCPM/RBBS). I can prepare draft documents on the go, then capture the incoming text file to disk.

A Basic program on the Kaypro (ADDLF.BAS) adds a line feed at each carriage return (required by CP/M and many non-TRS-80 computers). You can then print the resulting file as is, or process it further on your home computer. You can upload Model 100 Basic programs in a similar manner if you save them in ASCII format.

I use an Epson MX-80 printer. Its ability to skip over perforations by either software command or hardware switch lets you paginate documents created on the Model 100 without the intermediate step of uploading to a more powerful computer.

Anyone buying a printer for the Model 100 should look for this feature, as well as the capability to add line feeds (a hardware switch option on the MX-80).

The usefulness of the Model 100 is enhanced by the series of features initiated in your July issue. Although it is not my primary use of this computer, methods of installing machine-language code would be a good subject for a future article.

Phil Wheeler
5539 Towers St
Torrance, CA 9050[illegible]

Foxfighter Glitch

I just bought my first copy of *80 Micro*, and I'm delighted to find the C•Notes section dedicated to the Model 100. I am satisfied with my Model 100, but until recently thought that no one was writing software for it.

I enjoyed the Foxfighter program (August 1983, p. 200) but I did find a couple of glitches in it.

The program always presents one of seven predefined screen displays for the air mines. This becomes routine after a little while and encourages high scoring.

The changes shown in lines 10 and 25–34 in Program Listing 1 display the air mines in random patterns instead. Occasionally this results in an invisible air mine or two, adding to the challenge of the game.

Line 10 sets the RND function to one of 60 different starting points based on the Model 100's built-in clock. Lines 25–34 use the RND function to display the air mines in various screens instead of the seven predefined screens used in the original listing.

Also, line 150 of Foxfighter is supposed to provide an additional fighter plane when the score reaches 500, 1,000, and 2,000. Actually, it only increments the number of fighters displayed by 1 at these three points. Lines 150–158 in Listing 1 change the number of fighters as well as the display.

Harold Shaver
509 Mulberry #[illegible]
Suisun, CA 94585

```
10 DIMX(6),A(42):V=3:SCR=0:
CLS:PRINT@92,"<<FOXFIGHTER>>":
PRINT:PRINT:FORT=1TOVAL(RIGHT$(TIME$,2))
:SEC=RND(1):NEXT:INPUT"DO YOU NEED
INSTRUCTIONS (Y OR N)" ;A$:IFA$="Y"
THEN 390 ELSE 20
25 FORI=1 TO 42:READA(I): NEXT
30 FORM=1 TO 6
32 Y=INT(41*RND(1))+1:FORI=1 TO
6:IFX(I)=A(Y) THEN32
34 X(M)=A(Y):NEXTM
150 IFSCR=500 AND
BN=0THENV=V+1:BN=1:GOTO155ELSEIFSCR=1000
 AND BN=0THENV=V+1:BN=1:GOTO155ELSE IF
SCR=2000 AND BN=0THENV=V+1:BN=1:GOTO
155
155 IF SCR<> 500 AND SCR <>1000 AND SCR
<> 2000 THEN BN=0
158 PRINT@0,V
```

Program Listing 1. Adjustments for Brad Dixon's "Foxfighter" program.

Calculator Program

The short calculator program in Program Listing 2 is one of the first programs I wrote on my Model 100.

To run the program, input a value and press the enter key. Then input either a plus (P), multiplication (.), subtraction (–), or division (/) sign.

Type in your next number and press the enter key. Now press the equals key to get an answer, or key in another function and continue calculating. Once you have an answer, you can start over, stop, or carry your balance forward.

By using P for addition and a period for multiplication, you don't need to use the shift lever.

Mark Fox
774 Hazelwood Drive
North Wales, PA 19454

```
5 CLS
6 PRINT"CALCULATOR PROGRAM"
10 INPUT Y
11 CLS
15 PRINTY
20 Y$=INKEY$:IF Y$=""THEN GOTO20
21 IF Y$="="THEN100
25 PRINTY$
30 INPUT X
31 IF Y$="P"THEN Y=Y+X:GOTO20
32 IF Y$="-"THEN Y=Y-X:GOTO20
33 IF Y$="."THEN Y=Y*X:GOTO20
34 IF Y$="/"THEN Y=Y/X:GOTO20
100 PRINT"-------"
200 PRINTY;" IS YOUR ANSWER"
210 PRINT
220 PRINT"ANOTHER EQUATION? OR BAL
FWD(Y,N,B)"
223 Z$=INKEY$
230 IF Z$=""GOTO 223
230 IF Z$="Y"THEN5
240 IF Z$="Y"THEN5
250 IFZ$="B"THEN15
260 END
```

Program Listing 2. Model 100 Calculator program.

DiskCount Data™

WHERE QUALITY PROGRAMS MEET COMPETITIVE PRICES

2701-C W. 15th • SUITE 324 • PLANO, TX 75075 • (214) 680-8268

STOCKING STUFFER SALE!

All MOD III programs specified in this ad will run on the Model 4 (in the MOD III mode).

ULTRATERM
by United Software

OVER 15% OFF

The Ultimate Communications Package at a Very Reasonable Price

Retail 59.95
Special Sale **49.95**

This communications package is one of the most versatile available. It includes a full featured intelligent terminal program, host program, direct file transfer function, and hex conversion utilities for bulletin board downloading plus ● Support for virtually all manual and auto dial modems ● Exclusive Ultra-Term direct disk file transfer function allows full control from the transmitting computer, and unattended operation at the receiving computer ● Split screen feature lets you type outgoing messages on the bottom half of the screen while incoming messages are displayed on the top of the screen ● Full Line printer support ● Universal ASCII format file transmission and reception with memory input/output buffer supports delays between characters, delays between lines, and prompt controlled transmission for compatability with virtually all systems. A counter continuously displays the amount of free memory remaining whenever the buffer is in use ● Full featured host program that supports PRINT and PRINT graphics from basic and automatically relocates and protects itself in the top of memory is included ● Auto Dial phone numbers stored in program with single Key Auto Dial

MOD I or MOD III

POWERDRIVER $29.95
By Powersoft

NEW PRINTER DRIVERS

This new generation of custom printer drivers allows you to utilize all of SUPERSCRIPSIT'S features with your EPSON (Graftrax Required), PROWRITER and C.ITOH F-10 STARWRITER printers. You can now utilize all of the printers custom features such as compressed, expanded and proportional print, underlining, bold-face, super-scripting, and sub-scripting. All drivers can be called from within SUPERSCRIPSIT at "document open time." Order by printer designation as follows:

POWERDRIVER E	EPSON MX70/80/100
POWERDRIVER P	C.ITOH PROWRITER
POWERDRIVER S	C.ITOH STARWRITER

MOD I or MOD III

SUPER UTILITY PLUS VERSION 3.2
By Powersoft

30% SAVINGS

SU+ 3.2 79.95
Inside SU+ 19.95
Retail 99.90
SALE . . . 69.95

NEW SUPER UTILITY+ VERSION 3.2
Includes Operators Manual and 2nd Backup Disk
The Book Inside SU+ 3.0 Included FREE
A 19.95 Value at no extra charge
Experience a legend with the program voted as the outstanding utility of 1982.

MOD I or MOD III Protected Media

THE HOME ACCOUNTANT™
By Continental Software

An Outstanding Financial Planner

ONLY 71.99

● Maintains up to 100 budget categories ● Keeps track of up to five checkbooks ● Prints checks, if desired ● Prints a personal balance statement, income and expense summary ● Prints net worth statement ● Provides fast bank reconciliation ● Allows file extensions on multiple diskettes ● One program handles cash, checkbooks, credit cards and other liabilities and expenses ● Unlimited annual transactions, fiscal or calendar year ● Transactions may be "split" among different budget categories ● Flags transactions for tax purposes ● Maintains transaction history ● Provides Hi-Res graphics for any category by bar graph.
"The program itself does just about everything you'd ask of a personal finance package." -- Popular Computing, November, 1982

MOD III

INFOSCAN
By The Small Systems Center

Retail 49.95
Special 44.95

NEW ITEM

InfoScan is an information utility that is used to organize, store and retrieve any type of useful information in any format.
It combines the features of a word processor, data base manager, and information retrieval and display utility. You compose information records on the screen using the word processing feature, in any format you like. Each record is described by 1-17 character key phrase that you specify. Once created, the information file's keyphrases can be scanned as a list, on the screen, using an impressive key scan bracket. You scroll the keyphrases thru the brackets with the arrow keys on your keyboard. When the desired Keyphrase is found, you press a single Key and the corresponding information record is retrieved from disk and displayed on the screen. The record displays in exactly the same format as you used to create it. You can even design your own forms to "fill in".
A stunning visual presentation, it utilizes the 'display window' technique. Menus, Keyphrase lists, displayed text, and status information are displayed in a series of display windows on your screen. Impressive graphic display panels are used to enhance the visual orientation of the product and your information.

MOD I/III MINIMUM 48K 1 Drive

ZEUS
by Cosmopolitan Electronics

74.95

The newest editor/assembler from Vern Hester, author of Multidos, lays claim to being the fastest gun in the west. Once again Vern brings his magic to the TRS-80 with what may be the finest editor/assembler available. Will run with any DOS and contains all standard assembler features and then some. Fully documented, this is a great value.

Included is a mini disk operating system for the MOD I, III and 4.

MOD I/III/4

THE TEST GENERATOR
By Big G Software

ONLY 34.95

ATTENTION EDUCATORS AND STUDENTS

STUDENTS
This program is an ideal study aid for home computer use. A student types in review questions, vocabulary words, notes, etc. and the computer asks the questions. The student has two chances to answer correctly the the computer gives a score and shows questions answered incorrectly. The student then has the option to recycle incorrect answers.

TEACHERS
Type in up to 100 questions and answers to compose a test. The test may be printed in Multiple Choice, Matching, True-False, or Short Answer format. Questions may be printed in order or at random. Answers may be printed as questions, and questions as answers. An answer key is automatically printed for each test. Tests may be duplicated using Xerox, spirit masters, etc.

MOD III

READS ALL DOS SYSTEMS

SuperDirectory
By Computer Shack

Manufacturer's Suggested List 49.95
DiskCount Data Special 44.95

SAVE 10%

● **Build a Complete Library of All Your Programs**
● **Automatic Density Recognition**
● **Automatic Track Count Recognition**
● **Automatic DOS Recognition**
● **The Best Directory On the Market**

A great new directory program with its own operating system written by Vernon Hestor. This program will read any DOS there is, even compatability between MOD I and MOD III with doubler installed. The MOD I program can read most MOD III disks and the MOD III program can read most MOD I disks. 80 track double density is also supported.

DOSPLUS
by MicroSystems Software

SALE

SPECIALS ON ALL VERSIONS

VERSION	RETAIL	SALE
4 MOD 4	$150.	$129
3.5 MOD I or III	$150	$119
3.3 MOD I	$100	$44.95

Please Specify Model Number and Density When Ordering
Some Quantities Limited and Subject to Prior Sale

THE TOOLBOX for LDOS
by Powersoft

Previously 99.95
Reduced to 69.95

SPECIAL 59.50 SALE

SAVE 15%

PCHECK/CMD	PCOMPARE/CMD
PFIX/CMD	PCLEAR/CMD
PREFORM/CMD	PSS/CMD
PVU/CMD	PMAP/CMD
PERASE/CMD	PMX/FLT MX80
PMOVE/CMD	PHELP/CMD
PDIRT/CMD	PBOOT/CMD
PASSGO/CMD	PFILT/FLT
PUN/CMD	DVORAK/FLT
PEX/CMD	DVORAK/JCL
PMOD/CMD	CODE/JCL
PFIND/CMD	DECODE/JCL

MOD I or MOD III

CLEAN SLATE by Advanced Operating Systems

ONLY 49.95

RETAIL $79.95

A very unique word processing package, Clean Slate has many features not found in word processors at twice the price. It allows full customization of printer drivers and direct control of this printer from text. Features user definable repeat keys, non-destructive cursor, graphics print offset, custom keyboard driver and has it's own built-in comm program for file transmission and receipt. Can build a separate glossary file, and a names list for auto from letter insertion. You can also create and edit assembly source or basic program files. We found a treasure chest of this little known jewel at the right price. Two color documentation in a padded easle binder, it is an outstanding value for this price.

MOD I/III

EYE EASE GREEN CRT SCREEN FOR TRS-80 MODEL III

A MUST FOR THE SERIOUS COMPUTERIST

SAVE UP TO 40%
MOD I, II, III/4 only 14.95

- MACHINE BEVELED EDGES
- READY TO INSTALL
- SELF BONDING
- DUSTPROOF SEAL
- OPTICALLY CORRECT
- NO DISTORTION
- REDUCES EYE FATIGUE
- SHATTERPROOF
- ENHANCES IMAGE

$24.95

GREEN WINDOW

PREFERRED BY COMPUTER PROFESSIONALS EVERYWHERE

OPTICOM

6.0 PLUS
by Micro Systems Software

49.95

Enhance Your Model 4 TRSDOS 6.0 and Disk Basic

A great value from the DOSPLUS folks, these utilities for your MOD 4 DOS add some really important features. Enhancements include:

DISKZAP	MAP	REF
DISKDUMP	RESTORE	RESOLVE
DIRCHECK	SORT	BE1
	SR	BE2

MOD 4

POWERDOT II

NEW VERSION
RETAIL $59.95

The ULTIMATE in graphics design. This version is MUCH MORE POWERFUL than previous versions and includes BETTER documentation as well. New features include AUTODRAW™ and CIRCLE commands. Now allows you to design your own character sets! Includes lots of examples on disk. Your screen is only a "picture window" to a much larger drawing area! You are only limited by disk storage, not memory! Works on EPSON Series (Graftrax of Graftrax+ required) or the C.Itoh 8510 (PROWRITER or NEC version). Now available for DMP-2100, DMP400 and LP-8. SPECIFY PRINTER

SAVE OVER 15% NOW 49.95

LC COMPILER AND EDAS
By Misosys

SAVE OVER $60

LC Compiler Retail $100
EDAS Retail $100
SPECIAL BUY BOTH FOR $139.99

Now program in the "C" language compiler subset for LDOS. LC generates Z-80 EDAS IV source code as output. Compiled programs run on both MOD I and MOD III. Save $60.00 with the best LDOS editor assembler package on the market. Over 200 pages of documentation. Requires 2 drives and 48K.

FOR LDOS MOD I/III

GAMEPAK 3
by Micro Mining

ONLY 29.95

3 GAMES FOR THE PRICE OF ONE

This package by James Talley of Kid-Venture fame includes 3 games on the same disk. Funny Face is a program that draws over 1 million different faces on the screen as you select the different facial features. You can play "line-up" where you select the face from up to 6 like faces, or have the computer draw faces in the automatic mode. Matchmaker is a program like the TV game "Concentration" where you select identical graphics matches from a grid. It keeps a running point score. Repeat After Me emulates the famous "Simon" game where you have to enter a repeat sequence of numbers to match the computer. Sound included. Funny Face alone is well worth the price. Kids will adore this 3 game selection.

MOD I/III

MORE DISKCOUNTS

PROGRAM NAME	SPECIAL
BASIC S COMPILER SYSTEM I/III	39.95
MAKE 80 I or III	14.95
MICROTERM I or III or 4	69.95
SCRIPLUS 3.0 I/III	29.95
ST80 III I or III	124.95
DOSPLUS II II	199.95
MICROCASH III	149.95
M-ZAL ASSEMBLER RELEASE 2 I OR III	99.95
TALLYMASTER I or III	74.95
LDOS 5.1 I or III	119.95
SUPER UTILITY TECH MANUAL 3.)	11.95
INSIDE SUPER UTILITY+ 2.2z	15.95
THE CUSTOM TRS-80	26.95
HOW TO DO IT TRS-80	26.95
TRS-80 Disk & Other Mysteries	19.95
TRSDOS 2.3 Decoded	26.95
MICROSOFT BASIC DECODED	26.95
BASIC FASTER & BETTER	26.95
MACHINE LANGUAGE DISK I/O	26.95
INSIDE SUPER UTILITY+ 3.0	17.95

MULTIDOS
BY Cosmopolitan Electronics
Reads Other Operating Systems

VERSION 1.6 $99.95 Now 84.95
The latest release of an outstanding operating system
or
Buy VERSION 1.5
at an unbelievable
$49.95
MOD I or III SPECIFY

SPECIAL OFFER

Any order of over $100.00 or more from this ad will receive one choice of the following absolutely

FREE

- Green Window I/II/III/4
- Macro-Mon
  The Shadow

Any order over $200 will receive Clean Slate

FREE

POWERMAIL PLUS
By Powersoft
Retail 150.00
Special 112.50
SAVE 25%

Features Over the Original PowerMAIL

Separates any category for you in any manner you wish. This new version can span drives and may use multiple floppies. The program will sort on any eight levels if you wish. Separates your flags and puts them into another file. Print Options: Labels & Listing. One choice is the simple "telephone book" type listing. Two of the other modes give you not only the listing of names, but also either FLAG numbers, or the actual FLAG descriptions. The final print mode is what we call THE SALESMAN'S NOTEBOOK™. It prints out the whole list, or whatever parts you specify with all info including full flag descriptions in each record. Then the word "Notes" is printed with several lines of blank space for making those notes in the field"

MOD I/III/4 2/12/16
SPECIFY

LOAN AMORTIZATION
By Big G Software
ONLY 29.95

User enters principal, interest rate, and term of loan, and computer calculates monthly or annual level payment amount. An amortization schedule may be printed on request. Amortization schedule includes interest in current payment, principal in current payment, interest paid to date, principal paid to date, ending balance, and calendar year totals for interest and principal paid.

MOD III

SYSTEM DIAGNOSTIC
By Howe Software
ONLY $99.95

Complete diagnostic tests for components of your TRS-80. Separate tests for: ● ROM ● RAM ● VIDEO DISPLAY ● KEYBOARD ● LINE PRINTER ● CASSETTE RECORDER ● RS-232-C INTERFACE ● DISK DRIVES ● DISK CONTROLLER ● INDIVIDUAL TESTS AND CONTINUOUS SYSTEM TESTS

MOD I or MOD III

BBS-80
by MicroSystems Software
ONLY 74.95

Your computer becomes a mini communications network. BBS 80 allows a person to exchange messages or information via the telephone.
Access can be completely public, private to authorized users or limited to certain individuals.
BBS 80 can leave messages on another terminal acting as a rapid electronic mailman.
BBS 80 provides some of the most advanced and most convenient features ever incorporated in a microcomputer bulletin board system including:
Users may send private messages that may be read only by the addressee.
Multi-level system security allows up to 15 levels of security in both the message base and the upload download database.
BBS 80 "remembers" each regular user and stores terminal information, security access levels, last message retrieved during last call etc.
Individual user passwords.
Users may "upload" or transmit data and programs files to BBS 80.
These programs may then be "downloaded" to other BBS 80 users.
BBS 80 is self maintaining. Message space is automatically reclaimed when messages are deleted.

MOD I or III

PowerDRAW
by Powersoft
Retail 39.95
Special . . . 34.95
SAVE OVER 10%

A full screen graphics editor, POWERDRAW is 100% assembly language. You can create screens of graphics, save to disk, merge them, run in sequence like a movie, merge text with graphics, and write your own game or business application screens!
PowerDRAW saves graphics to disk or tape so that they can be recalled at a later time in the following formats.

0) CONDENSED TOKENS
1) EDAS Source file format
2) EDTASM Source file format
3) BASIC data statements
4) BASIC Strings
5) BASIC arrays (New in version 1.3!)
6) Load File Format

MACRO-MON THE SHADOW
BELOW DEALER COST

A HUGE DISKCOUNT DATA BUY PASSES THESE UNBELIEVABLE SAVINGS ON TO YOU.

Formerly Manufactured by
ADVANCED OPERATING SYSTEMS
Now a DiskCount Data exclusive

The industry's standard and the finest break point disassembler and machine language monitor available. The package includes a rich, padded easel binder and complete 2 color documentation. Now single step through your CMD files, view & change assembly op-codes, memory, etc. Used by Kim Watt for the editing of SuperUtility+.

MOD III Disk	99.95	Now Only	34.99
MOD I Disk	64.95	Now Only	32.99
MOD I Tape	59.95	Now Only	29.99

Quantities Limited & Subject to Prior Sale

CP/M 2.2
by Digital Research and Montezuma Micro

TRS-80 MODEL 4
Only 199.95

Now, run CP/M on your Model 4 and ur leash the powerful features resident in your computer. Take advantage of CP/N software such as WordStar®, dBASE II and Multiplan™, along with thousands of others.

● Includes INTERCHANGE™, a utility that allows read ng, writing and copying 20 different manufacturers' disk formats such as IBM, KAYPRO, OSBORNE, XEROX, etc. ● Includes MEMLINK™, a unique feature that uses the optional 64K RAM memory as a fast disk drive. ● Com lete with all these CP/M utilities: ASM, DDT, DUMP, ED, LOAD, PIP, STAT and SYSGEN. ● Includes MODEM 7, a powerful public domain communications program. ● Supports 80 x 24 video, reverse video, direct cursor a dressing and more. ● Utilizes the Model 4 function keys and allows user defined keys. ● FORMAT utility permit: up to 52 disk formats to be constructed, all menu driven. ● Ready to run in the standard 64K Model 4. The add ional, extra cost, 64K RAM upgrade not required.

MICROPRO SOFTWARE

The full line of MicroPro softwa e is now available formatted for the Model 4 using this CP/M.

WordStar® Fast memory mapped version	$250
MailMerge® Multi-purpose file merging program	125
SpellStar® 20,000 word proofreader on a disk	125
StarIndex™ Creates index and table of contents	95
WordStar Professional. All the above for only	450
InfoStar™ Advanced DBMS	250
ReportStar™ Report generator & file manipulator	175
DataStar™ Data entry and retrieval package	150
SuperSort® Fast and flexible sorting is yours	125
CalcStar™ Advanced electronic spreadsheet	95

PRO-PIX
By Micro Mining™
NFL Weekly Predictions
NOW 19.99

Up dated for 83-84 season, Pro-Pix features a unique scoring algorithm, along with other weighted probability factors. You input weekly scores to all prescheduled games (Full NFL schedule by week or by team is included in program) and watch Pro-Pix get smarter as season progresses. Returned over 62% win recommendation last full NFL year. Printer routines included; it will print win or lose predictions and point spreads for all teams weekly. Recreational use only. Should not be used for wagering.

MOD I or III

THE MASTER MECHANICS SET FOR LDOS By Powersoft
A 39.95 VALUE
NOW 34.99

9 UTILITY PROGRAMS TO ENHANCE YOUR LDOS LIBRARY

PMOD/CMD	DISK MODIFICATION UTILITY
PCHECK/CMD	DIRECTORY CHECK UTILITY
PFIX/CMD	DIRECTORY REPAIR UTILITY
PVU/CMD	SECTOR VERIFICATION UTILITY
PCLEAN/CMD	DISK CLEANUP UTILITY
PSS/CMD	FILE SECTOR STATUS
PREFORM/CMD	REFORMAT WITHOUT ERASE
PMAP/CMD	DISKETTE AND FILE MAP
PASSGO/CMD	PASSWORD REMOVAL

MOD I or MOD III

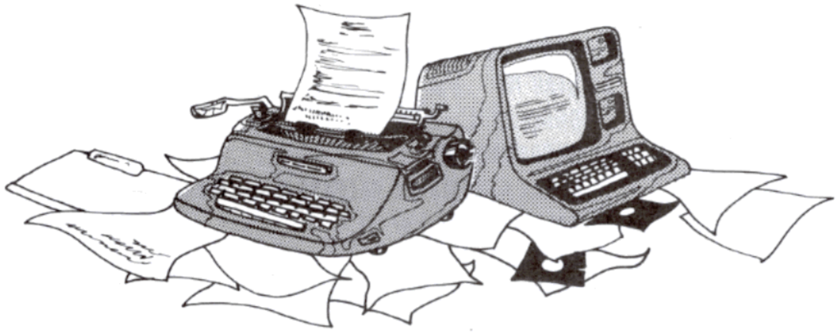
DiskCount Data TM

214-680-8268

Phone Your Order In Today Or Mail To
DISKCOUNT DATA
2701-C W. 15th St., Suite 324
Plano, TX 75075
Office Hours Mon-Fri 10A.M. to 9P.M. C.S.T.
Send Cash, Check or Money Order
Please add $3.00 for postage and handling additional $1.50 for C.O.D.'s.
Foreign orders welcome, please specify air or surface. All shipping charges assumed by purchaser.
When ordering by mail, please specify computer model number (I, II, or III), drive configuration, and memory size.
AND
Cheerfully Accepted
✓204

✓ See List of Advertisers on Page 291

NEWS THIS MONTH



14,000 miles with a Model 100

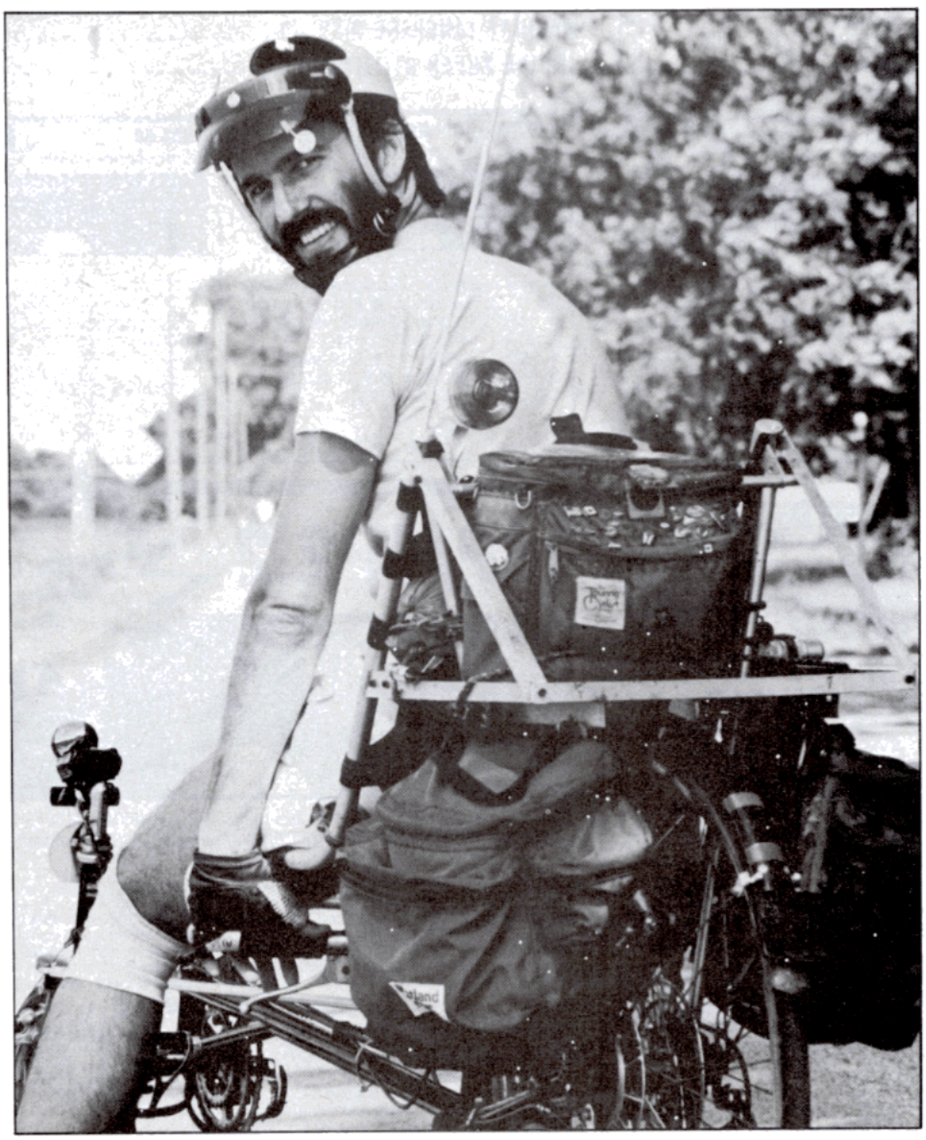
Steve Roberts' bicycle odyssey.

Anyone who's bought a TRS-80 Model 100 will tell you that the portable lets you work without being chained to a desk, but one owner is taking Tandy's "Micro Executive Work Station" idea far beyond an armchair or plane ride. Steve Roberts, a Columbus, OH, freelance writer, plans to conduct a year's business with his Model 100, without stepping into his office once.

In fact, most of the time he'll be lying down—on a custom-designed recumbent bicycle, festooned with generators and solar cells, which he'll pedal 14,000 miles across America.

On the road, Roberts will record his various writing projects using a helmet-mounted microphone and a portable tape recorder. After setting up camp, he'll transcribe his prose onto his 32K Model 100 and upload it via telephone to his main computer, a Micromax System 1000, in Columbus. Kacy Branstetter, Roberts' manager and editor, will be standing by to receive copy, forward phone calls, and serve as what Roberts calls his "interface with the universe."

Roberts' .DO files, he says, will make a pit stop between bike and Branstetter: The trip's primary sponsor, CompuServe Information Service, is supplying "essentially unlimited time and file-space" for uploading. CIS members are encouraged to chat with the rolling writer (his user ID number is 70007,362), and his saga will be available as a regularly updated data base: "I think you'll be able to just type GO SKR and it'll be there as a display file," Roberts predicted.



Roberts: "I'll exist in a totally asynchronous fashion."

An on-line travelogue isn't Roberts' first high-tech writing assignment. A former software and systems consultant, the 31-year-old author turned to freelancing in the late 1970s. As head of Words'worth Inc., a business communications firm, he alternates magazine ar-

ticles and books (Prentice-Hall's *Creative Design with Microcomputers* and *Complete Guide to Microsystem Management*) with corporate technical writing assignments. The former, he told CompuServe's *Today* magazine editor Carole Gerber, offer "fame and glory," while the latter "[provide] steady income."

His bicycle trip is a shot at both. Interviewed two weeks before his scheduled Sept. 28 departure, Roberts told *80 Micro* he planned a 700-mile "shakedown cruise" around Indianapolis, IN, and Louisville, KY. Then, in early November, he'll begin his journey with a turn south from Washington, DC, toward Florida to start a clockwise loop of the U.S.

"I'm anticipating [the trip's taking] about a year, finishing up around the idea, of course, is to take advantage of as much good weather as possible."

In preparation for the voyage, Roberts said, he was wringing out both his Model 100 and his legs: "I'm basically living off the Model 100 now for everything. Between the Model 100 and CompuServe, my office is very light."

As for physical preparation, "The obscene part of all this is that I've been riding around central Ohio, which is very flat and [where it's] very easy to get cocky about your ability to go long distances. [But] unless there are physical problems I don't see any trouble. I may bust a knee and jettison all this high-tech equipment."

He's already jettisoned an item that would have spared him frequent visits to phone booths. Technical problems and the unit's weight aborted his plans to carry a second computer, a homemade CMOS CP/M system with 3½-inch microfloppies for mass storage; that leaves him the stock Model 100, no storage except RAM, and obligatory stops to upload every 15 pages or so.

"I'm hoping to get something that'll be useful for [bulk storage] fairly soon," Roberts admitted. "I've looked at wafertape drives, and I'm interested in what people are doing with 3½-inch floppies. If all else fails I have cassette, though I'm not crazy about it.

"If anyone comes up with more memory, I'm ready. I've talked a little to Holmes Engineering about the 256K bubble memory and it sounds like it's not quite ready, but that would be a worthwhile purchase if it comes out sometime during the trip."

Meanwhile, he estimates that his cycle, camping gear, and electronics equipment total about 110 pounds, some taken up by a high-tech way to save batteries: "A couple of solar panels that were donated by Solarex provide 15 volts in full sun," he said, and another sponsor has contributed rechargeable Ni-Cad cells.

Besides relying on them, plus an ac line when available, Roberts has a generator similar to those that run bicycle headlights: "If I've got a good tailwind or I'm flying downhill, it's a simple matter to pop [the generator] against the wheel and get power."

There's one item he doesn't expect to use—a CB radio, carried in case of emergency. "I've stripped it down almost to just a PC board," Roberts said. "It's not even near me when I'm doing my regular writing, but if I'm crashed in a ditch somewhere I'd like to be able to call for help."

While the trip will challenge not only Roberts' stamina but the Model 100's—"how well it will do in the sun and vibration and dirt and everything else"—Radio Shack is keeping its distance from the project. "They know about it but seem kind of unresponsive," Roberts said when asked about any contact with Tandy. "I encountered such a huge, faceless organization when I [approached them] for a sponsorship that I went back to CompuServe."

That may be because Radio Shack is unwilling to share other people's publicity stunts. Roberts seems sincere and enthusiastic about the trip and eager to discuss it as an affirmation of new technologies' ability to liberate deskbound workers, but the enterprise is not exactly free of Madison Avenue-style merchandising.

In addition to his CompuServe updates, there's the matter of what Roberts will write during his year on the road. Besides freelance magazine articles and material for corporate clients, he plans two books. One will be either a "boring but easy to write" text about on-line communications and engineering, or a computer science text which Roberts calls "potentially a huge money-maker but a lot of work."

Roberts' second and more important effort will be a book tentatively titled *Computing Across America: A Bicycle Odyssey*, for which his agent is currently negotiating with several publishers.

Suspicion of Roberts' making the trip in order to write about it, rather than making it and then writing about it, lessens the credibility of his words to *Today*'s Gerber: "The whole trip offers an opportunity to test the viability of the information society. I want to see if I can maintain a heavily interactive, information-oriented professional practice involving a lot of clients [with] complete freedom from the confines of an office. I'll exist in a totally asynchronous fashion."

Also, Roberts' point is to be free from desks and papers, yet he'll be carrying generators and solar cells. Is that practical? "I think it is practical," he told *80 Micro*, "and I've been doing it on a very small scale recently as I've been practicing.

"I find I get a lot more done when I'm out on a beach or something with a Model 100. This morning, in fact, I wrote most of an article at a Wendy's restaurant while having breakfast. When I'm out of the office, there are fewer distractions and I can get more done."

This makes sense, but sounds more like most users' appreciation of the 100 as a handy notepad than a defense of a full-time career with one. Also, of course, Roberts' job fits his thesis better than most other professions would; the freelance writer is the most insecure financially, but the least place- and equipment-bound, worker there is.

Nevertheless, Roberts insists, his trip goes beyond both commercial aspects and his point about the open office. "If I wasn't writing for a living and had lots of money I'd be doing it anyway," he said. "The fact that I'm not independently wealthy forces me to work while I'm on the road, and [the 100's] a convenient way to do that.

"I don't think it is [a publicity stunt]. I've thought about it a lot and wondered if it was, and it doesn't feel that way. It's something I do privately as well as publicly."

And, anyway, Roberts' tour is an adventure. *Computing Across America* is unlikely to rival de Tocqueville's *Democracy in America* or William Least Heat Moon's *Blue Highways*, but Roberts' combination of gee-whiz Woodstock spirit and "Real People" PR might produce a bestseller.

How about *Zen and the Art of Model 100 Maintenance*? ■

—*E.G.*

✓ See List of Advertisers on Page 291

SOFTWARE

ROM is copyright

Apple wins; Franklin to appeal.

Diving into a hazy maze of 0's and 1's, a federal appeals court in Philadelphia has issued a ruling that makes the question of software copyright one step clearer, even as it blurs the distinction between software and hardware.

On August 30, a three-judge panel upheld Apple Computer Inc.'s lawsuit against Franklin Computer Corp., allowing Apple to seek an injunction against sales of Franklin's Apple-compatible Ace 1000. The key decision: computer manufacturers can copyright operating systems and programs in ROM, as well as applications programs.

Writing that "the medium is not the message," Circuit Court Judge Darcy Sloviter overturned a lower court's denial of Apple's suit. Franklin admitted copying 14 operating system programs before the lower court, but argued that such programs embodied in chips are essential parts of the machine—hardware—and, as such, ineligible for copyright.

The presiding federal district judge agreed, finding that firmware was not written "in a language of description" and refusing to issue a temporary restraining order against Franklin. The appeals court's decision lets Apple return to district court and seek the injunction, even as Franklin's lawyer, James Shestack, announced plans to ask for a rehearing—round 3 of the battle, so to speak—before the entire Court of Appeals. Round 4 might take place in the U.S. Supreme Court.

In upsetting Franklin's claim that the programs were an uncopyrightable "process, system, or method of operation," Sloviter and his colleagues said that that approach "mistakenly focuses on the physical characteristics of the instruction," like paying attention to a book's ink and paper rather than its contents. "Apple," the panel declared, "does not seek to copyright the method which instructs the computer to perform its operating functions but only the instructions themselves.

"Franklin's attack on operating system programs as 'methods' or 'processes' seems inconsistent with its concession that application programs are an appropriate subject of copyright," the court continued. "Both types of programs instruct the computer to do something.

"The statutory definition of a computer program...makes no distinction between application programs and operating programs. We reaffirm that a computer program in object code embedded in a ROM chip is an appropriate subject of copyright."

The decision makes an important distinction between the two legal means by which people protect their ideas—copyright and patent. As the *New York Times*' David E. Sanger wrote, "Under U.S. law, copyrights protect the expression of an idea, such as a literary work. Ideas themselves, in the form of novel inventions, are protected by patents."

In Franklin's view, Apple's operating programs were unpatented hardware, and therefore free for copying. Computer makers, Sanger pointed out, "have shied away from using the patent system to protect their programs," because patents take a long time to obtain and because "it is not clear whether most computer programs are sufficiently novel and distinct from one another to merit patent protection."

For example, Scripsit and Newscript, both TRS-80 word processors, are alike in many ways—generally, they're both written in 1's and 0's; more specifically, they use similar routines to perform similar tasks such as opening and closing files. Like two novels written with the same words, they are not different enough to be patented. They are, however, copyrighted by Radio Shack and Prosoft respectively.

As for Apple, the Cupertino, CA, firm's vice president and general counsel, Albert Eisentat, was naturally pleased with the ruling, telling *Computerworld*, "I think it's one of the most definitive statements of the law that's been done yet."

Whatever the odds, however, Franklin vowed to continue the fight. The Cherry Hill, NJ, company's executive vice president and chief operating officer, Avram Miller, told reporters, "Our plans right now are to go back to the court for redress. We believe we'll prevail and the injunction will be denied."

Added attorney Shestack, "We still contend that Apple was abusing the copyright laws to gain a monopoly on

As this issue went to press, Osborne Computer Corp. stopped production of its portable computers, laid off 300 of its 400 remaining workers, and filed for protection from creditors under Chapter 11 federal bankruptcy laws.

The Hayward, CA, firm employed 1,000 people before closing its New Jersey plant and beginning California layoffs last summer, plagued by more powerful and less expensive competitors to its Osborne 1 and delays in shipping its successor, the Osborne Executive.

On Sept. 12, two San Jose component suppliers filed a lawsuit claiming Osborne owed them more than $4.5 million for circuit boards, disk drives, and other parts. The portable pioneer filed for Chapter 11 protection in U.S. Bankruptcy Court in Oakland on Sept. 13.

James Lopes, attorney for Osborne, told the Associated Press that three banks had agreed to loan the company $600,000 while Osborne sought possible buyers or investors.

80micro's REVIEW GUIDE

The guide that will save you time and money.

You'll spend more time at your computer and less money for the right product.

80 Micro's Review Guide is the most comprehensive collection of TRS-80* reviews ever compiled in book form. You get 500 hardware, software and book reviews packed with information about:

- Modems
- Games
- Business programs & Word Processing
- Joysticks
- Books on programming
- Printers
- Utilities
- Editor/Assemblers
- Educational software
- Monitors
- Databases
- And more

Don't waste valuable time chasing down the right product. We've done it for you with **80 Micro's Review Guide**, compiled and condensed from the pages of **80 Micro**, the magazine you've come to rely on.

A five-star rating system lets you decide what's best. And each review gives you the product manufacturer's information so you can order and receive your product *fast*.

All for $7.95

Order now. Use the attached order form, the coupon, or call toll free 1-800-258-5473.

**TRS-80 is a trademark of Radio Shack, a division of Tandy Corp.*

Outside USA, Phone: 603-924-9471

yes, I want to save time and money.

Send me **80 MICRO's REVIEW GUIDE** today.

Please send me ______ Review Guides at $7.95 each and add $1.50 per book shipping and handling.

80RGD

☐ Check enclosed ☐ MC ☐ VISA ☐ AE

Please allow 6-8 weeks for delivery

Card #______________________ Exp. Date__________

Signature ______________________

Name ______________________

Address ______________________

City______________________ State______ Zip Code__________

Wayne Green Inc. • 80 Pine St. • Peterborough, NH 03458 **Attn: Marcia Stone**

equipment compatible with its machine."

Ekos Inc., a Franklin distributor, issued a press release affirming its support of the Ace manufacturer. Ekos President Steven R. Gerbsman declared, "There are other alternatives not yet explored, including royalty payments, that cause us to believe we'll be marketing Franklin computers for some time to come."

The mention of royalty payments indicates a possible direction for settlement, though no one's talking at present; all Miller would tell *ISO World* was, "We have contingency plans if we fail."

One of Franklin's contingency plans is probably its IBM PC-compatible micro, being prepared under the direction of engineer William Sydnes, who designed the original PC and reportedly was working on IBM's Peanut when he moved to Franklin in April.

The Philadelphia ruling does not affect makers of IBM clones, because IBM's "open architecture" policy makes details of its PC system readily available.

Nevertheless, the Apple/Franklin decision means that computer companies can protect all software—whether on a disk, a listing, or a chip, and in source or object code—by copyright. Manufacturers continue to worry about small-scale piracy and product backups, but system pirates have apparently been sunk. ■

—*E.G.*

BUSINESS

Waiting for Santa

The micro industry looks to the Christmas market.

After a summer of rollercoaster stock prices and grim predictions for TI and Atari, the microcomputer industry tried to pull itself into shape for the Christmas buying season. By September, some companies had asserted themselves as movers and shakers, some were on the move, and others were merely shaking.

Taking the latter first, **Texas Instruments** moved swiftly to reduce towering TI 99/4A inventories. A price cut to $99 helped move the overstocked machine, and a peripheral package consisting of a disk drive and controller, 32K RAM expansion, and rack mount, was slashed from $1,200 to $550. In addition, stores received a free $100 software package for every system ordered.

According to *Electronic News*, dealers immediately started selling the expansion package and software for as little as $449, with industry observers predicting $399 before long. Such a price, bringing a 48K disk system to around $500, would be competitive, though other questions remained unanswered: whether buyers would be attracted to the software-scarce micro, and whether TI could survive at such a low (perhaps negative) profit margin.

Turning the knife in TI's wound, Montgomery Ward issued its 1983 Christmas catalog on September 9—and bumped the 99/4A to give space to Coleco's Adam.

Timex dealers had even less to smile about. Ken Coach, marketing and sales director of Softsync, told *InfoWorld*, "The Timex-Sinclair 1000 has petered right out." Coach hoped that the forthcoming TS 2068, the U.S. version of the Sinclair Spectrum color machine, might share the under-$200 market with Commodore, but saw no prospects for the black-and-white TS 1500: "I'm not putting any hopes on it at all."

Following TI's lead, meanwhile, **Atari** cut prices for most of its lineup, from the 2600 VCS game console (effectively trimmed to $59 after a rebate) to the new 600XL and 800XL home computers, given wholesale tags of $140 and $240 respectively. Even so, Atari's retail prices were some $60 above the competition's—the 600XL versus the 99/4A in the 16K arena, and the 800XL versus the Commodore 64. Just as TI disappeared from Montgomery Ward shelves, the 203-store Target chain added Coleco and dropped Atari.

The Warner Communications subsidiary seemed to be pinning its hopes on its AtariSoft line of programs for non-Atari micros, announcing 12 games for the 99/4A and eight for Commodore, IBM, and Apple. In addition to computer versions (expected to retail for $38 to $50 apiece), Atari launched several titles for rival Intellivision and ColecoVision game machines.

While VIC-20 sales slowed to a crawl, **Commodore** 64's were jumping off the shelves. While $199 prices gave only a $5 margin over wholesale cost, vendors relied on peripherals and software for profit. For instance, 90 percent of C64 buyers also choose the 1541 disk drive—supplies of which nearly ran out in late summer, obliging Commodore to schedule an emergency airlift from Japan.

Spinnaker Software President Bill Bowman echoed the industry consensus when he told *InfoWorld*, "If the low end is going to have a savior, it will be the Commodore 64." The only machine that seemed competitive was **Radio Shack**'s white-cased 64K Color Computer, which debuted at $399.95—well under the C64's original $595, but twice Commodore's current price. (The compact new Color Computer 2 with 64K RAM and Extended Basic lists for $468.95, plus upgrade installation.)

How low will Commodore go? In its July 25 issue, *ISO World* claimed that building a C64 costs Commodore less than $60 and that the firm could "apparently sell the 64 for $99 wholesale and still make a profit."

While no one seemed ready to tackle Commodore in the trenches, many manufacturers were comfortable in the higher levels of the market. **Kaypro** was making about 12,000 machines a month, while **Apple** and **IBM** each produced perhaps 70,000 IIe's and PC's respectively. **Compaq** predicted $100 million in revenues for its first year, selling 50,000 copies of its portable PC clone.

There were rumors of more **middle-range micros** to compete with Adam and Peanut—a $500, 16-bit, 256K Commodore; an under-$1,000 TI with concurrent CP/M in ROM. Steve Wozniak told the *San Jose Mercury News* that Apple's McIntosh, due in November or December for $1,200 or so, would "just [boggle] everyone. It's just totally unbelievable and unexpected, not just a better version of something that is already around."

Coleco rebounded from pessimists' gossip to show production units of Adam (though a press release said "less than $700" instead of the usual $600 cost figure), and Child World, Markline, and Diners Club joined Wards and Target in placing orders for fall delivery. In mid-September, though, the press reported that Adam's FCC approval would be a month behind schedule.

Coleco also joined **AT&T** in plans for "an interactive game and entertainment service," bringing Zaxxon and company to anyone with a home computer or game console. The service will use standard phone lines and an AT&T/Coleco modem; subscribers will pay a monthly charge plus user fees to play a game. The announcement upstaged Mattel, which developed the Playcable service now offered by 20 cable TV companies.

And **IBM** prepared to upstage everybody, with 100,000 Peanuts sold by Christmas. There was talk of still other news from Big Blue—a portable PC by year's end, a $10,000, 32-bit challenger to Lisa by spring 1984—but IBM's home computer continued to rule the gossip world.

Analysts couldn't agree on specifications, *Datamation* reported, but they agreed on one thing: Whether it cost $700 or $900, had 64K or 90K, and used a tape drive or a standard 5¼-inch disk or IBM's orphan 3.9-inch disk or CP/M or MS-DOS on a ROM chip, Peanut would use the IBM name to become the dominant force in home computing.

In the words of Peter Cunningham, president of Input, a Mountain View, CA, research firm: "Coleco's Adam might be state of the art for the home market and have the lowest retail price, but it will probably be IBM that makes all the money." ■

—E.G.

PULSETRAIN

Magic/L challenges Basic

WorkSlate: Designed to do one thing well.

Loki, in Norse mythology, is a malevolent, mischievous god, always making trouble around the gods' home of Asgard and (in Marvel Comics' version) trying to defeat the mighty Thor. If Basic is the Thor of microcomputer languages, Loki Engineering of Cambridge, MA, is hoping to raise a little mischief of its own with a syntax called Magic/L.

According to *Mass High Tech*, Magic/L is "a Forth equivalent with simpler syntax and a high level of interactivity with the user." Its authors, Loki's Jeff Epstein and Arnold Morris, compare it to C and Pascal in terms of structure, but claim it's more powerful and more interactive.

"Magic/L is much faster than Basic, and can be learned in hours," Morris said. "[It's] the ultimate in user friendly, and once you get good, it allows you to do things like access Assembly language directly."

While working on satellite data display systems at the Smithsonian Astrophysical Laboratory, Epstein recalled, he and Morris became "frustrated because [Forth] was so hard to read. Forth is what some people call a write-only language." In other words, what looks fine to a Forth programmer might prove baffling to someone who wants to modify the program later.

In addition, Epstein claims, Forth and other languages "have their roots in prehistory. Basic, for heaven's sake, was written on an IBM 1130, much less of a machine than an Apple."

To overcome these shortcomings, the two programmers developed Magic/L, working on a Data General Nova in Epstein's basement in early 1981. Today, *Mass High Tech* says, Loki "is betting its future that Magic/L will become a standard language of the 1980s—perhaps even competing head-to-head with Basic in the personal computer market."

After selling it as a development tool for minicomputers, Marketing Manager Barry Unger said, Loki planned to launch CP/M-86 and MS-DOS versions of the new language at Boston's CP/M East show in October, and hoped to make Magic/L a household word before long.

"New languages are usually developed by large teams at universities," Unger said, "but we believe we have something that is way ahead of anything else, and we see a tremendous growth potential."

Who knows? If Magic/L proves a threat to Microsoft and Digital Research, Loki Engineering may incur the wrath of the gods.

Non-programmable portable

While Tandy's Model 100 and its rivals are lap-sized workstations suitable for almost any computing task, Convergent Technologies Inc., of Santa Clara, CA, has taken a different approach. Its WorkSlate is designed for specific applications, giving executives a spreadsheet, calculator, terminal, and telephone in an 8½- by 11-inch package.

The 3½-pound portable, introduced in ComputerLand and Businessland stores and in the American Express catalog, has a 16-line, 46-character LCD display, an 8-bit CMOS 6800 CPU, a 300-baud modem, and a built-in microcassette recorder. Its 16K RAM is expandable to 32K, but its strength is its 64K ROM—which includes an advanced spreadsheet and windowing capabilities.

For instance, users can put a spreadsheet at the top of the display and a financial calculator at the bottom, moving data back and forth between the windows.

WorkSlate's audio recorder lets it serve as a speakerphone and phone-answering machine; spreadsheet templates, called Taskware, are available on special data/voice tapes. Its maker, Convergent's new Advanced Information Systems division, claims users can use the tape recorder to make vocal annotations to spreadsheets, up to 10 of which can be stored on a microcassette.

The firm plans communications, financial-modeling, and memo-writing software—not full-featured programs, but patches or templates to WorkSlate's spreadsheet environment. There may also be different machines for writers and students; Convergent marketing manager Karen Toland admits Work-

See List of Advertisers on Page 291

Slate's circular keys discourage typing, but says, "Executives don't type. They do word processing with their mouths.

"WorkSlate is designed to replace the calculator, pencil, paper, and eraser," Toland told *InfoWorld*'s John Markoff and David Needle. "It will not compete directly with the Tandy Model 100."

The unit costs $895, with Taskware tapes selling for $19.95 to $49.95 and a printer for $250.

Small is profitable

Most office automation firms such as DEC, Wang, and IBM hope to place their products with the prestigious Fortune 1,000 companies, whose business totals over $1.8 trillion in annual revenues. However, according to Focus Research Systems of West Hartford, CT, there's an even bigger market among small businesses. U.S. companies with annual sales of under $25 million apiece combine for over $2 trillion a year—and the advent of office micros has made them prime candidates for automation.

American business, Focus points out, consists "of a handful of large companies and a gigantic number of small businesses": of an estimated 3.56 million nonagricultural companies, 3.55 million—99.7 percent—have under 500 employees. Perhaps 3.1 million, or 88 percent of all U.S. businesses, have fewer than 20 people on their payrolls.

Except for a copier (most firms with eight or more employees have a copy machine), these offices stayed unautomated while mainframe and mini salesmen made their rounds. "A small company with $200,000 in annual sales may allocate $4,000 annually for automation, and such a sum hardly warrants a major sales effort by any vendor," Focus' survey, "Small Business Automation," admits.

The microcomputer era changed that. From 1979 to 1982, Focus' survey states, 254,000 small firms bought computers. Of those buyers, 80 percent had 1–19 employees; while only 78,000 firms that size had a computer in 1979, over 278,000 had one three years later.

Looking ahead, Focus predicts over 900,000 small businesses will buy their first computer within the next three years. By 1984–85, the researchers claim, the computer market among companies with under 450 workers will total $8 billion, with more than half of that coming from firms with under 20.

"Computer vendors with low-cost products and distribution systems capable of selling in volume are well positioned to reap the benefits of this massive market," Focus concludes.

Say it with paper

In a recent "Side Tracks" column (November 1983, p. 6), *80 Micro*'s editor-in-chief Eric Maloney remarked that the magazine prefers submissions and queries on paper to messages on CompuServe. The market researchers at International Resource Development Inc. see broader implications, describing an anti-videotex backlash that illustrates Hegelian philosophy.

Proclaiming, "The 'Paperless' Home? According to Hegel, No Way!", IRD claims that the telecommunications boom will actually increase the consumption of certain kinds of paper—that, with "uniformity, mechanization, and depersonalization" as the thesis, people will long for its antithesis.

"People don't need paper just for business reasons," says IRD researcher Ken Bosomworth. "They need paper for personal reasons, too. A letter written on personalized stationery will be far more meaningful than the same words appearing on a CRT—and, for that matter, more meaningful than the same words printed out on computer paper."

Thus, the Norwalk, CT, analysts predict, the future will bring not one big happy Network Nation but "a resurgence in demand for" high-quality stationery, business forms, and greeting cards, as people seek more sincere correspondence. "And the Hegelian synthesis," IRD concludes, "shall be a world in which the old and new media are each appreciated for the respective strengths they bring to communications."

While rejecting the idea of videotex Valentines, Bosomworth admits that telecommunications will prevail over printed catalogs, directories, and Yellow Pages. "The synthesis-antithesis concept only applies in situations where deeply felt human needs are involved," he says. "If it's a question of efficiencies on the one hand and no emotional need on the other, the efficiencies will win out every time."

CRT users see pink

It seems the debate over possible health effects of CRT displays will never end. A National Academy of Sciences panel concluded in July that terminal use has no adverse effects on operators' vision, but the National Institute of Occupational Safety and Health (NIOSH) vowed to continue research on health-related CRT problems, including a study focusing on CRTs and pregnancy (see *80 Micro*, October 1983, p. 294).

Now three IBM PC users have discovered that gazing into a computer monitor does indeed have an effect on vision, if not exactly an adverse one. Look at a CRT long enough, and white figures on a black background turn pink.

Susan Greenwald, an Evanston, IL, architect, noticed the color distortion after a session of PC word processing. Greenwald's husband Mark, an ophthalmologist, contacted Randolph Blake, a Northwestern University psychologist who specializes in visual perception. Blake took one look at the CRT and gave a rosy diagnosis: the McCollough effect, a minor optic maladjustment that can last from several minutes to several weeks.

"I routinely explain the McCollough effect in my lecture courses," Blake told *Computerworld*. "The conditions under which the effect is produced have to be fairly constrained. But it just so happens that the green letters on the black terminal generate just this aftereffect.

"The most interesting thing is that the pattern of the color distortion conforms to the pattern and contour of the letters on the screen. That is, if you stare for a long time at a CRT, then see white letters on a black background that are of similar size and shape, you'll get the pinkish tint."

While the effect is physiologically harmless, Blake told *CW*'s Katherine Hafner, physicians should be aware of it in case computer-owning patients call, worrying that they've suddenly been struck color-blind.

To former typists or scriveners, the McCollough effect may be a small price to pay for the convenience of word processing ("There aren't really a lot of instances where we read white letters on black," Blake admitted). If you'd rather see straight, time away from the CRT and keyboard will restore normal color perception. ■

Uncle Sam goes micro

END BYTES

● News flash: The **FEDERAL GOVERNMENT** has discovered that microcomputers are powerful tools, if people know how to use them. Halfway through a six-month study designed to develop policies and plans for federal micro use, the General Services Administration told *Computerworld* that employees find micros help them make decisions faster and complete their work more quickly and accurately.

"In-depth, hands-on training and ongoing availability of technical assistance are essential," the GSA's preliminary findings declare. "Also, formal instruction in software selection and application are essential in managing the successful transition to microcomputer use."

● Radio Shack, continuing its new policy of **COOPERATION** with outside software marketers, has swapped conversion rights with CBS Inc. Tandy earns worldwide rights to market TRS-80 versions of selected CBS programs, while CBS will sell some Radio Shack software for non-Radio Shack micros. The agreement marks the first time Tandy has allowed its software to be converted to other formats.

● As reported in October's End Bytes (p. 300), Tano Corp. of New Orleans, LA, has brought the **DRAGON** to the U.S., one year after its debut in the U.K.—where it's ceased to impress the British magazine *Computer Dealer*. "The Dragon is beginning to look distinctly archaic," writes *CD*'s Peter Craig. "Sales figures prove the value of good marketing and distribution over a good product."

Assessing the low-end micro market in Britain, Craig comments that Apple's IIe "still suffers from having the decimal point in its price in the wrong place" and that its "specification is also beginning to look a bit dated," while "the soon-to-be-released Atari models are disappointing evolutions of the old models." Commodore, Craig concludes, is "arrogant in knowing that the 64 is a world beater. The machine is so cheap to manufacture and so superior in performance that Commodore could quite easily zap their competition from a vast altitude."

● From distributing printers and disk boxes, **LEADING EDGE PRODUCTS** has moved into the IBM PC market. The Canton, MA, firm has followed its much-ballyhooed PC word processor with its own MS-DOS micro, an 8088-based system said to run 70 percent faster than the PC and cost 40 percent less. The 128K machine, assembled by a number of overseas contractors, will be the first hardware product marketed under the Leading Edge name.

● 1983 was the year that computer prices fell through the floor; 1984 may be the year that **SOFTWARE PRICES** follow. Microsoft Vice-President Jim Spillars told *ISO World* in July, "I think you are going to see products in the $100 to $150 range by Christmas that have been selling in the $250 to $350 price range."

By September, Spillars' prediction seemed on target, with Commodore cutting some C64 software prices 50 percent. Silicon Valley Systems of Belmont, CA, slashed its popular Apple word processor, Word Handler, from $199 to $59.95, and offered a package of Word Handler and List Handler, formerly $298, for $89.95. It's clear that buyers of the new sub-$1,000 micros are unwilling to pay $250 apiece for programs.

● Despite extra courses and upgraded facilities, *The Wall Street Journal* reports, America's **COLLEGES** are falling behind the demand for computer classes. Georgia Tech has had to impose quotas on computer science majors, and the University of Wisconsin at Madison turns away 1,000 would-be computer science students a semester. Those who get in may have to use a terminal located in a hallway outside a crowded classroom, with terminal time available only after midnight.

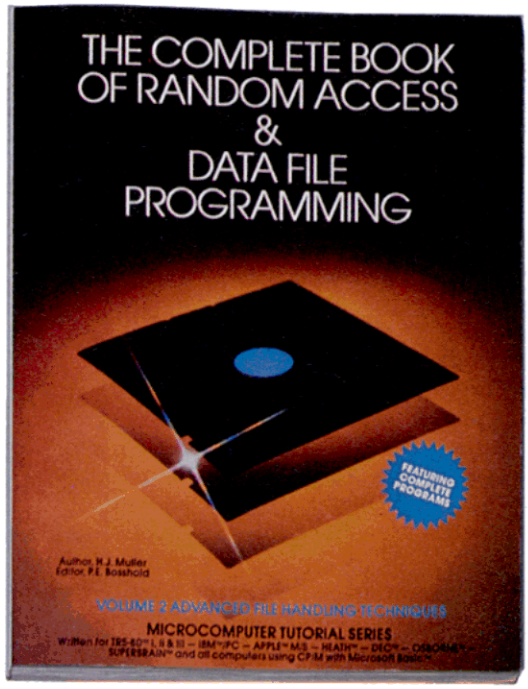
● The Model 100 is only the beginning: The market researchers at International Resource Development Inc. see **PORTABLES** as accounting for one-quarter of all personal business computers and office workstations by 1987. More and more portables will have integrated voice/data functions like Convergent Technologies' WorkSlate's, that can answer the phone and digitize users' spoken notes as addenda to files or programs; by the late 1980s, "pocket consultants" with optical-card memory should replace "whole shelves of medical or law books" for professionals in those fields.

● The Boston-based analysts of the Yankee Group, meanwhile, see **TELECOMMUNICATIONS** as achieving "mass market status in the fourth quarter of 1985." By then, a Yankee study predicts, 20 percent of home computers and up to 12 percent of video game consoles will have modems, and an additional 1.5 million telephones will sport terminal capabilities and built-in displays.

● And, if all those on-line homes grow tired of talking to each other, they can turn to The Source and do **CROSSWORD PUZZLES**. The McLean, VA, data base now offers a weekly British-style (cryptic clues) puzzle, created by New York attorney J. Baxter Newgate. Besides having "instant access to the answers" and being able to challenge or compliment the author, Source puzzlers will soon have the option of choosing different levels of difficulty. ■

Learn to Program Like a Professional!

THE COMPLETE BOOK OF RANDOM ACCESS & DATA FILE PROGRAMMING



Written for TRS-80™ I, II, & III - IBM™/PC - APPLE™ M/S - OSBORNE™ - HEATH™ - DEC™ - SUPERBRAIN™ - and all Computers using CP/M with Microsoft BASIC™

The last word on disk random access and file handling techniques, this series is intended for everyone — beginning programmers, businessmen and professionals will learn how to create custom programs to handle inventories, mailing lists, work scheduling, record keeping, and many other tasks, while more experienced programmers will learn advanced, professional programming techniques for faster, more efficient data storage and retrieval.

Although random access file handling is a matter of some complexity, the subject has been treated in a simple and down-to-earth fashion, so that anyone with some small familiarity with programming in Microsoft BASIC will be able to cope with the material. Each stage of learning uses a sample program as a starting point. The programs grow in capability and complexity as the books progress into all of the various aspects of file handling and record manipulation. An extensive effort has been made to keep the material coherent and every program line is explained in detail.

Volume I
BASIC FILE HANDLING

- The writing of a Menu to summarize program functions
- Screen format for data entry
- The creation of a basic record
- The FIELD and LSET routines for buffer preparation
- Disk storage of random access records
- Changing or editing stored records
- The LPRINT capability from disk using three different formats
- Sorting the random file
- Searching by name or key field
- Search in "next" or "prior" fashion
- Purging deleted records
- Using disk file data for calculations
- Future expansion of data fields
- Using flags to prevent program crashes
- Date setting, printer on-line and many other routines to make a program run like a commercially written program

VOLUME I **$29.95**
Option Vol. I Program Disk
TRS-80 Model I/III **$28.50**
TRS-80 Model II **$32.50**

Volume II
ADVANCED FILE HANDLING TECHNIQUES

- Relational database programming
- Comprehensive self-balancing accounting system with printouts
- Hashcoded data file manipulation -- (probably the fastest method of data retrieval). Hashing the input key and recovery method explained
- Span-blocking techniques allow creation of records longer than 256 bytes without wasted space
- Blocking & Deblocking
- Shell-Metzner sort
- In-place screen editing
- Recovery of deleted record space
- Alpha-index record retrieval
- Fast machine language/BASIC sort
- Linked list record structure and sort-merge, deleted record removal and file reorganization
- Multi-key file reorganization and record searching

VOLUME II **$29.95**
Optional Vol. II Program Disk
TRS-80 Model I/II/III **$49.95**

COMPUTRONICS INC.
50 N. PASCACK ROAD
SPRING VALLEY, NEW YORK 10977 ✓9

NEW TOLL-FREE ORDER LINE (OUTSIDE OF N.Y. STATE)
DEALER INQUIRIES WELCOME
(800) 431-2818

24 HOUR ORDER LINE
(914) 425-1535

30-DAY MONEY BACK GUARANTEE

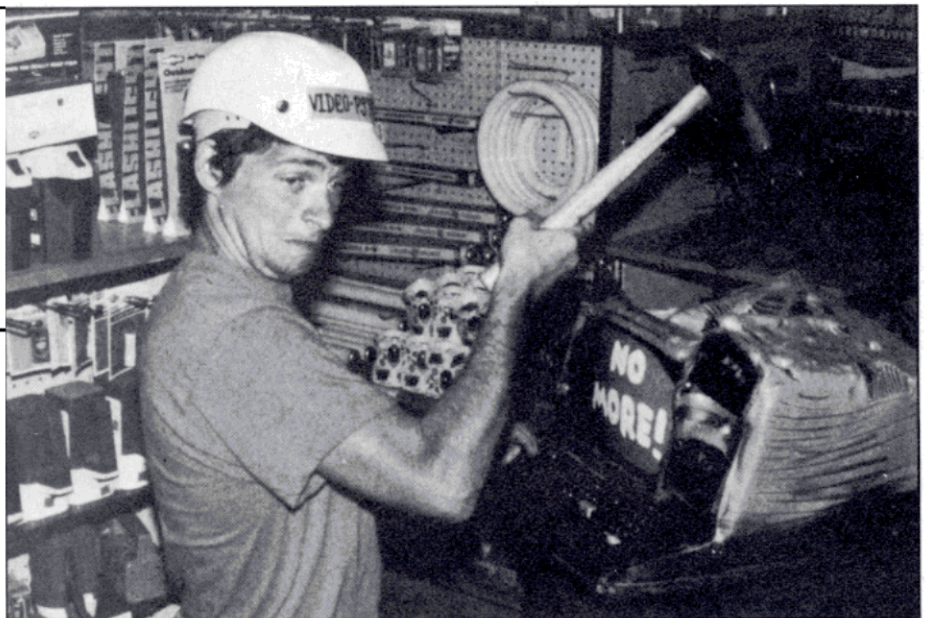
*** ALL PRICES & SPECIFICATIONS SUBJECT TO CHANGE ***
DELIVERY SUBJECT TO AVAILABILITY

★ ADD ★
ADD $3.00 FOR SHIPPING IN UPS AREAS
ADD $4.00 FOR C.O.D. OR NON-UPS AREAS
ADD $5.00 TO CANADA & MEXICO
ADD PROPER POSTAGE OUTSIDE OF U.S., CANADA & MEXICO

The GAMER'S CAFE

by Rodney Gambicus

Notes from New England



(Editor's note: With regret and some embarrassment, we must report that the Gamer's Cafe did not submit a column this month. We're not exactly sure why, since Rodney and Mad Max were right here in New Hampshire under our watchful eye. They disappeared mysteriously one day without even a goodbye, leaving behind only a photo of some guy smashing a computer and a disk marked "Misc." That disk contained, among the clutter of half-finished programs and bizarre bits of prose signed "T. DeQuincey" (we suspect that Max wrote them, owing to the many references to the Woodstock Nation and Studebakers), a Scripsit file entitled "Notes," which we suspect comprised the core of the December column. Those scrawls (and it's not easy to scrawl in Scripsit) are presented here, minus, of course, the many spelling errors and coffee stains.)

Whine, whine, whine! It seems like everybody is accusing everybody else of cheating on Big Board games. We're going to need the Magnificent Seven to clean up the mess. "It's no fun reading the Big Board mail anymore," says Max.

For starters, Scott Trent challenges Jer McLanahan's 261 in Space Warp. "If you have ever played that game one of the first things you will notice is that the highest possible score is 255," he claims. "The other thing is that it is very

The Big Board

Apple Panic	287,620	Mary Phinney, Stockbridge, MI
Arex	138,780	Rob Mitchell, Peterborough, NH
Assault	97,457	Zagros Sadjadi, Petaluma, CA
Astroball	317,240	Stefan Kunze, Moers, W. Germany
Attack Force	1,732,820	Dave Smith, Raleigh, NC
Bable Terror	8,857	Mad Max
Barricade	17,520	Troy Scrapchansky, Uncasville, CT
Caterpillar	362,883	Alvah Werner, New Albany, OH
Centipedes	94,836	Belinda Chron, Tempe, AZ
Chicken	12,035	Noble Chowchuvech, Demarest, NJ
Clash	174,300	Zagros Sadjadi, Petaluma, CA
Convoy	34,770	Rick Sayre, Stockton, CA
Cosmic Fighter	806,280	Robert Newman, Stoney Creek, Ont.
Crazy Painter	1,087,000	Mike Beebe, Sacramento, CA
Cyborg	317,000	Robert Cavin, Laredo, TX
Danger in Orbit	69,640	Steve Sustacek, Danube, MN
Defense Command	128,230	Bette Dufraine, Bolton, CT
Demise/Defend	165,000	David Russell, Ardrossan, Scotland
Demon Seed	103,160	Markus Blum, Ludwigshafen, W. Germany
Desert Peril	84,400	Jay McLain, Clatskanie, OR
Devil's Tower	25,700	Rick Sayre, Stockton, CA
Dungeon Escape	6,531	Donald Tindall, Littleton, CO
Flying Saucers	2,186	Stuart Lory, Victoria, B.C.
Fortress	515,925	Greg Samson, Loudonville, NY
Frogger	400,900	Shawn Roberts, Oklahoma City, OK
Galactic Empire	1,850	David Russell, Ardrossan, Scotland
Galaxy Invasion Plus	1,600,058	Shawn Lipman, Nelspruit, S. Africa
Gauntlet	58,360	David Schwartz, San Jose, CA
Ghost Hunter	41,190	John Kane, Nelson, N.Z.
Insect Frenzy	691,156	Tommy Seniuk, Vegreville, Alta.
Invaders from Space	655,360	Darren Cotter, Oceanside, CA
Jovian	148,300	Greg Samson, Loudonville, NY
Jungle Boy	851,900	Zagros Sadjadi, Petaluma, CA
Killer Gorilla	28,312	Alex Poon, Baton Rouge, LA
Laserball	72,530	Neil Matson, Panama City, FL
Laser Defense	1,504,610	Greg Samson, Loudonville, NY
Leaper	144,500	Tommy Seniuk, Vegreville, Alta.
Lunar Lander	15,100	Brent Lewis, Long Valley, NJ
Mad Mines	10,220	Gorman Miller, Titusville, FL
Martian Patrol	30,496	David Schwartz, San Jose, CA
Meteor Mission 2	119,750	Bob Brown, Dallas, TX
Missile Attack	44,000	Raimo Hansen, Mesa, AZ
Monster Invaders	32,620	Troy Scrapchansky, Uncasville, CT
Olympic Decathlon	10,278	Adrie van Geffen, Rotterdam, Netherlands
Outhouse	1,000,000	Kyle Hoyt, Titusville, FL
Panik	85,075	Mark Owens, Houston, TX
Penetrator	585,460	David Schwartz, San Jose, CA
Planetoids	56,450	Carl Pflanzer, Gillette, NJ
Rear Guard	195,240	John Hope, Kingston, Ont.
Robot Attack	143,250	Mark Fertig, Northville, MI
Sea Dragon	610,180*	Robert Fitzwilliam, Houston, TX
Sky Sweep	1,000,540	Tommy Seniuk, Vegreville, Alta.
Space Castle	69,750	Rick Sayre, Stockton, CA
Space Intruders	14,030	Ron Johnston, Emporia, KS
Space Warp (Level 8)	261	Jer McLanahan, New Canaan, CT
Stellar Escort	625,000	Kevin Josephson, Chilliwack, B.C.
Super Nova	2,138,710	Mark Fertig, Northville, MI
Swamp Wars	59,130	Farhad Abrishami, Silver Spring, MD
Temple of Apshai	390	Carl Pflanzer, Gillette, NJ
Time Runner	89,479	Mad Max
Venture	58,550	Darren Cotter, Oceanside, CA
Voyager 1	833	Farhad Abrishami, Silver Spring, MD
Weerd	61,180	Tommy Seniuk, Vegreville, Alta.
Wild West	15,400	Gorman Miller, Titusville, FL

*Expert mode: 339,080 (David Smith, Kingwood, TX).

H&E COMPUTRONICS INC.

• • EVERYTHING FOR YOUR TRS-80™ • ATARI™ • APPLE™ • PET™ • CP/M™ • XEROX™ • IBM™ • OSBORNE™ • •

* TRS-80 is a trademark of the Radio Shack Division of Tandy Corp. · * ATARI is a trademark of Atari Inc. · *APPLE is a trademark of Apple Corp. · * PET is a trademark of Commodore
* CP/M is a trademark of Digital Research · *XEROX is a trademark of Xerox Corp. · * IBM is a trademark of IBM Corp. · * OSBORNE is a trademark of Osborne Corp.

NEW

★ All orders processed within 24-Hours
★ 30-Day money back guarantee

BUSINESS PAC 100

100 Ready-To-Run Business Programs

(ON CASSETTE OR DISKETTE).....Includes 128 Page Users Manual.....
Inventory Control.....Payroll.....Bookkeeping System.....Stock Calculations.....
Checkbook Maintenance.....Accounts Receivable.....Accounts Payable.....

BUSINESS 100 PROGRAM LIST

	NAME	DESCRIPTION
1	RULE78	Interest Apportionment by Rule of the 78's
2	ANNU1	Annuity computation program
3	DATE	Time between dates
4	DAYYEAR	Day of year a particular date falls on
5	LEASEINT	Interest rate on lease
6	BREAKEVN	Breakeven analysis
7	DEPRSL	Straightline depreciation
8	DEPRSY	Sum of the digits depreciation
9	DEPRDB	Declining balance depreciation
10	DEPRDDB	Double declining balance depreciation
11	TAXDEP	Cash flow vs. depreciation tables
12	CHECK2	Prints NEBS checks along with daily register
13	CHECKBK1	Checkbook maintenance program
14	MORTGAGE/A	Mortgage amortization table
15	MULTMON	Computes time needed for money to double, triple, etc.
16	SALVAGE	Determines salvage value of an investment
17	RRVARIN	Rate of return on investment with variable inflows
18	RRCONST	Rate of return on investment with constant inflows
19	EFFECT	Effective interest rate of a loan
20	FVAL	Future value of an investment (compound interest)
21	PVAL	Present value of a future amount
22	LOANPAY	Amount of payment on a loan
23	REGWITH	Equal withdrawals from investment to leave 0 over
24	SIMPDISK	Simple discount analysis
25	DATEVAL	Equivalent & nonequivalent dated values for oblig.
26	ANNUDEF	Present value of deferred annuities
27	MARKUP	% Markup analysis for items
28	SINKFUND	Sinking fund amortization program
29	BONDVAL	Value of a bond
30	DEPLETE	Depletion analysis
31	BLACKSH	Black Scholes options analysis
32	STOCVAL1	Expected return on stock via discounts dividends
33	WARVAL	Value of a warrant
34	BONDVAL2	Value of a bond
35	EPSEST	Estimate of future earnings per share for company
36	BETAALPH	Computes alpha and beta variables for stock
37	SHARPE1	Portfolio selection model-i.e. what stocks to hold
38	OPTWRITE	Option writing computations
39	RTVAL	Value of a right
40	EXPVAL	Expected value analysis
41	BAYES	Bayesian decisions
42	VALPRINF	Value of perfect information
43	VALADINF	Value of additional information
44	UTILITY	Derives utility function
45	SIMPLEX	Linear programming solution by simplex method
46	TRANS	Transportation method for linear programming
47	EOQ	Economic order quantity inventory model
48	QUEUE1	Single server queueing (waiting line) model
49	CVP	Cost-volume-profit analysis
50	CONDPROF	Conditional profit tables
51	OPTLOSS	Opportunity loss tables
52	FQUOQ	Fixed quantity economic order quantity model
53	FQEOWSH	As above but with shortages permitted
54	FQEOQPB	As above but with quantity price breaks
55	QUEUECB	Cost-benefit waiting line analysis
56	NCFANAL	Net cash-flow analysis for simple investment
57	PROFIND	Profitability index of a project
58	CAP1	Cap. Asset Pr. Model analysis of project
59	WACC	Weighted average cost of capital
60	COMPBAL	True rate on loan with compensating bal. required
61	DISCBAL	True rate on discounted loan
62	MERGANAL	Merger analysis computations
63	FINRAT	Financial ratios for a firm
64	NPV	Net present value of project
65	PRINDLAS	Laspeyres price index
66	PRINDPA	Paasche price index
67	SEASIND	Constructs seasonal quantity indices for company
68	TIMETR	Time series analysis linear trend
69	TIMEMOV	Time series analysis moving average trend
70	FUPRINF	Future price estimation with inflation
71	MAILPAC	Mailing list system
72	LETWRT	Letter writing system-links with MAILPAC
73	SORT3	Sorts list of names
74	LABEL1	Shipping label maker
75	LABEL2	Name label maker
76	BUSBUD	DOME business bookkeeping system
77	TIMECLCK	Computes weeks total hours from timeclock info.
78	ACCTPAY	In memory accounts payable system-storage permitted
79	INVOICE	Generate invoice on screen and print on printer
80	INVENT2	In memory inventory control system
81	TELDIR	Computerized telephone directory
82	TIMUSAN	Time use analysis
83	ASSIGN	Use of assignment algorithm for optimal job assign.
84	ACCTREC	In memory accounts receivable system-storage ok
85	TERMSPAY	Compares 3 methods of repayment of loans
86	PAYNET	Computes gross pay required for given net
87	SELLPR	Computes selling price for given after tax amount
88	ARBCOMP	Arbitrage computations
89	DEPRSF	Sinking fund depreciation
90	UPSZONE	Finds UPS zones from zip code
91	ENVELOPE	Types envelope including return address
92	AUTOEXP	Automobile expense analysis
93	INSFILE	Insurance policy file
94	PAYROLL2	In memory payroll system
95	DILANAL	Dilution analysis
96	LOANAFFD	Loan amount a borrower can afford
97	RENTPRCH	Purchase price for rental property
98	SALELEAS	Sale-leaseback analysis
99	RRCONVBD	Investor's rate of return on convertable bond
100	PORTVAL9	Stock market portfolio storage-valuation program

☐ TRS-80 Cassette Version	$99.95
☐ TRS-80 (Mod-I or III), Pet, Apple or Atari Versions	$99.95
☐ TRS-80 Mod-II, IBM, Osborne and CP/M Versions	$149.95

ADD $3.00 FOR SHIPPING IN UPS AREAS
ADD $4.00 FOR C.O.D. OR NON-UPS AREAS
ADD $5.00 TO CANADA AND MEXICO
ADD PROPER POSTAGE OUTSIDE OF U.S., CANADA AND MEXICO

NEW TOLL-FREE ORDER LINE
(OUTSIDE OF N.Y. STATE)
(800) 431-2818

H&E COMPUTRONICS INC.
MATHEMATICAL APPLICATIONS SERVICE™
✓9

50 N. PASCACK ROAD
SPRING VALLEY, NEW YORK 10977

24 HOUR ORDER LINE
(914) 425-1535

ASK FOR OUR 64-PAGE CATALOG
DEALER INQUIRIES INVITED

ALL PRICES & SPECIFICATIONS SUBJECT TO CHANGE
DELIVERY SUBJECT TO AVAILABILITY

✓ See List of Advertisers on Page 291

hard to play Space Warp if you have other games." Maybe that's why 261 has stood for so long.

Then there's this cranky letter from Greg Samson of Loudonville, NY. He doesn't trust anybody, including the high-score holders for Martian Patrol, Cosmic Fighter, and Robot Attack. And he openly accuses Venture king Darren Cotter's 58,550 of being phony, charging that you can't get any more than 57,500. That Samson is going to be a real popular guy; I hope he can keep his kneecaps intact.

Max says all this cynicism concerning high scores is because of the 1968 Democratic Convention. He walks around the *80 Micro* offices shouting, "The whole world is watching! The whole world is watching!"

I was hoping this vacation would do him some good, but I don't know.

* * * * *

The one-line games keep trickling in. Most of them come from Australia, where the magazine seems to arrive three to five years late. I think I could swim there with a copy in my teeth and beat the overseas mail.

* * * * *

The van finally broke. We've got it down at Roland's Exxon for a tune-up. Max wanted to replace the plugs with pennies, but I think that only works in fuse boxes.

* * * * *

I tore Max away from Convoy long enough to get his opinion on dropping some more scores from the Big Board.

"PURGE," he grunted.

"Dig Out (Y/N/Q)," I said. "Y." "Paddle Pinball (Y/N/Q)." "Y." "Scarfman (Y/N/Q)." "Y."

You have to know how to speak Max's language if you want to get through to him.

* * * * *

Confessions of an Honest Gamer: Jack Martin of Somis, CA, sent in a score of 999,970 in Scarfman with the comment, "I feel honor bound to mention that there is a weird mode in Scarfman which enables one to obtain an unlimited number of additional men and that is the way in which these high scores are obtained. Playing with the customary number of men, the highest score obtainable is probably less than 400,000."

Confessions of a Sneaky Kind of Guy: James Griffith of Searcy, AR, reports 910,980 and says, "This score was achieved by a secondary methodology which I prefer not to disclose yet."

"Well, nah-nah to you, too," Max commented snidely.

* * * * *

Today, I found a blueberry on Mount Monadnock the size of a volleyball.

* * * * *

Max is nuts about Computer Shack's Convoy. He particularly likes the suicidal paratroopers. They run backwards into your truck and then explode.

"This is almost as good as a George Romero movie," he keeps saying.

* * * * *

Maybe we should retire Lunar Lander, too.

"In the [September] Gamer's Cafe you stated that the final disgrace will be in the cheating of Lunar Lander," writes Seth Eliot of Brooklyn, NY. Then he proceeds to give instructions on refueling:

On the Model I, exit the game by pressing Break and Reset, hit Enter at the memory size prompt, type in SYSTEM, and then type in /33767.

"It seems that this memory location refuels you without any damage to the game," writes Seth. "Sorry, Max."

"Not half as sorry as I am," Max grunted.

* * * * *

"It's obvious to me that you have a severe problem with very high scores in the Gamer's Cafe," writes Mark Schmidt. "Why don't you print the scores players get by just using one man?"

I thought it was an interesting suggestion, but Max was less than excited.

"Sure," he said. "Credibility? Verification? No problems."

* * * * *

Shawn Lipman's father called Peterborough twice in one day to report Shawn's Galaxy Invasion Plus score. We don't take high scores over the phone too often (partly because the van doesn't have a phone), but the Lipmans live in Nelspruit, South Africa. Wonder how long it takes the mail to get there?

* * * * *

Max is depressed. First it was the Big Board mail. Then Mary Schmidt of Stockbridge, MI, wrote, "Tell Mad Max to watch out; I'm within a few points of knocking him off his Bable Terror perch."

"Oh, yeah?" Max shouted. "*Oh, yeah?*"

Finally, to end the week, we were about to post Max's high score in Convoy when Rick Sayre of Stockton, CA, blew him out with 34,770.

"Oh, yeah?" Max mumbled meekly as he slumped over the keyboard.

* * * * *

The guys at Wally's Hardware in Spencer, MA, sure know how to have a good time.

"These people are *strange*," Max marveled as he looked at a photo of some fellow smashing a Model III with a large hammer.

"When it comes to serious gaming, we employees at Wally's Hardware are real hard-core players," said Bob Noonan in the accompanying letter. "We wring out every point we can get, until the computer screams for mercy."

If they're not playing games, Bob and Dave (the goofy stockboy) are reformatting all of Wally's disks or copying Galaxy Invasion on top of his accounts receivable program.

"Don't tell Wally that Dave cut the cord off the plotter to fix Mrs. Archambeault's lamp, 'cause he didn't feel like wiring in a new cord cap from stock," Bob added, "or that I chopped up his IBM/360 Assembly language textbook to show a customer how a gasoline-powered Weed-Whholloper worked."

"These people are *really* strange," Max repeated.

* * * * *

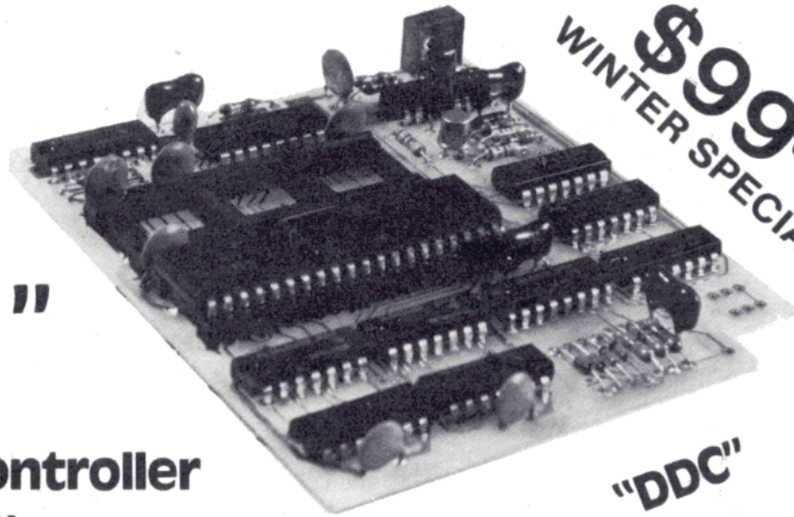
Max is restless. He wants to hit the road again. I've suggested that we go down to Boston to visit Mercedes while she works on her top-secret project at MIT.

The van is fixed. The guy at Roland's called it a miracle. He suggested that we change the oil at least once every 50,000 miles.

I don't know if I'm going to have time to write the December column. Oh, well—I'm sure the people at *80 Micro* will figure something out. ■

Aerocomp's Proven Best-By Test! The "DDC"

Double Density Controller



★ Technical Superiority

At last! A double density controller for Model I with HIGHER PROBABILITY OF DATA RECOVERY THAN WITH ANY OTHER DOUBLE DENSITY CONTROLLER ON THE MARKET TODAY! The "DDC" from Aerocomp. No need to worry about the problems that keep cropping up on existing products. AEROCOMP'S new analog design phase lock loop data separator has a wider capture window than the digital types currently on the market. This allows high resolution data centering. The finest resolution available with digital circuitry is 125 ns (nano seconds). The "DDC" analog circuit allows infinately variable tuning. Attack and settling times are optimum for 5-1/4 inch diskettes.
The units presently on the market use a write precompensation circuit that is very "sloppy". Board to board tolerance is extremely wide - in the order of ± 100 ns. The "DDC" is accurate to within ± 20 ns.
The bottom line is state of the art reliability!

★ Test Proven

Tests were conducted on AEROCOMP'S "DDC", Percom's "Doubler A"* and "Doubler II"* and LNW's "LNDoubler"** using a Radio Shack TRS80*** Model I, Level 2, 48 K with TRS80 Expansion Interface and a Percom TFD100 * disk drive (Siemens Model 82). Diskette was Memorex 3401. The test diskette chosen was a well used piece of media to determine performance under adverse conditions. The various double density adapters were installed sequentially in the expansion interface.
The test consisted of formatting 40 tracks on the diskette and writing a 6DB6 data pattern on all tracks. The 6DB6 pattern was chosen because it is recommended as a "worst case" test by manufacturers of drives and diskettes. An attempt was then made to read each sector on the disk once - no retrys. Operating system was Newdos/80, Version 1.0, with Double Zap, Version 2.0. Unreadable sectors were totalled and recorded. The test was run ten times with each double density controller and the data averaged. Test results are shown in the table.

★ Features

TRS80 Model I owners who are ready for reliable double density operation will get (1) 80% more storage per diskette, (2) single and double density data separation with far fewer disk I/O errors, (3) single density compatibility and (4) simple plug-in installation. Compatible with all existing double density software.

SUMMER SPECIAL $99.00
for the Best DD Controller on the market.

"DDC" and LDOS **$169.95**
"DDC" and Newdos 80 **$179.95**

★ TEST RESULTS ★

MFR & PRODUCT	SECTORS LOCKED OUT (AVG)
AEROCOMP "DDC"	0
PERCOM "DOUBLER II"	18
PERCOM "DOUBLER A"	250
LNW "LNDOUBLER"	202

Note: test results available upon written request. All tests conducted prior to 8-25-81

Aerocomp's 14 day money back guarantee applies to hardware only. Specials will be prorated. Shipping $2.00 in Cont. U.S. See opposite page for details. Add $4.00 shipping & handling for DDC & DOS.

Data Separators

The advances that make the "DDC" great are incorporated in the new AEROCOMP Single Density Data Separator ("SDS") and Double Density Data Separator ("DDS").

★ Has your original manufacturer left you holding the bag?

If you already own a Percom "Doubler A", "Doubler II" or LNW "LNDoubler" or Superbrain, the AEROCOMP "DDS" will make it right. Look at the test results:

MFR. & PRODUCT	SECTORS LOCKED OUT	
	WITHOUT "DDS"	WITH "DDS"
PERCOM "DOUBLER II"	18	1
PERCOM "DOUBLER A"	250	0
LNW "LNDOUBLER"	202	0

Note: Same test procedures as "DDC".
* Trademark of Percom Data Co.
** Trademark of LNW
*** Trademark of Tandy Corporation

★ "DDS"..... $49.95
(Use 1791 chip from your DD Controller)

★ "DDS" with disk controller chip included....... $79.95

★ Disk controller chip..... $34.95

(Shipping $2.00 Cont. US - see opposite page for details)

Do you need a Single Density Data Separator?

The internal data separator in the WD1771 chip (R/S Expansion Interface) is NOT recommended by WD for reliable data transfer. Do you have any of these problems: Lost data, tracks locked out, CRC errors, disk retry? YOU NEED ONE!

★ "SDS"..... $29.95
(For Mod. I; shipping $2.00)

Plugs directly into your existing Double Density Controller.

See opposite page ▸▸▸▸▸▸

DISK DRIVES

40 & 80 TRACK

SINGLE & DOUBLE SIDED

as low as **$169**

NEW!

PACESETTERS

Aerocomp leads the way to the BEST value in disk drives on the market. Quality, performance, reliability, warranty, service plus free trial – that's what you get from the leader. AEROCOMP!

BEST FEATURES

★ Fast 5 ms. track-track access
★ Single or double density
★ Easy entry door
★ "Flippy" feature allows read-write to the back of the diskette to cut media cost in half! (MPI)
★ Disk ejector (MPI)
★ External drive cable connection (no need to remove the cover to hook up the cable)

NEW!

MODEL III & 4 CONTROLLER and DRIVES

Convert cassette Model III or 4 to disk. Complete internal drive kits with 40 track SS drives (Tandon), Aerocomp disk controller board (will take up to 4 drives), power supply, mounting towers, all hardware & cables.

★ DRIVE KIT (no drives) $199
★ ONE DRIVE SYSTEM $369
★ TWO DRIVE SYSTEM $539

Shipping & handling $8.00 per system

★ AEROCOMP DISK CONTROLLER ... $119

Shipping & handling $2.00 per controller

★ MOUNTING KIT w/o Drives & Controller .. $99

Shipping & handling $4.00 per kit

COMPLETE DRIVES

TRS80 Mod. I & III, IBM PC & TI 99/4A. Power supply & enclosure. Specify silver or almond. 5.25 inch.

★ 40 track single side (Tandon) $199
★ 40 track SS "Flippy" (MPI) $239
★ 40 track Dual Head (either) $279
★ 80 track SS (MPI) $299
★ 80 track SS "Flippy" (MPI) $329
★ 80 track Dual Head (Tandon) $379

Shipping & Handling $5.00 per drive.

BARE DRIVES

Internal drives for TRS80 Mod. III, IBM PC, TI 99/4A, 5.25 in. (controller required)

★ 40 track Single Side (Tandon) $169
★ 40 track Dual Head (either) $249
★ 80 track SS (MPI) $269
★ 80 track Dual Head (Tandon) $339

Shipping & Handling $4.00 Per Drive.

8 INCH DRIVES

Drive expansion box complete with power supply and fan. Tandon Slimline.

★ Two (2) 8" Single Side $699
★ Two (2) 8" Double Side $849
★ 8" Bare Slimline, SS $260
★ 8" Bare Slimline, DS $375

Shipping & Handling $5.00 Per Drive.

MODEL I STARTER PACKAGE

One 40 track SS drive, 2-drive cable, TRSDOS 2.3 disk & manual, freight & insurance (Tandon).

$249

MISCELLANEOUS GOODIES

★ TRSDOS 2.3 disk & manual $20
★ LDOS (Mod. I or III) $119
★ NEWDOS/80, 2.0 (Mod. I or III) $129
★ Diskettes (10 in library box) $23
★ 5.25" Drive Power Supply & case $59
★ 2-Drive Cable $24
★ 4-Drive Cable $34
★ Extender Cable $13

Shipping & Handling $2.00

PERSONAL CHECKS WELCOME

We'll be happy to accept your personal check with any mail order without any shipping delay.

FREE TRIAL OFFER

Use your AEROCOMP drive for up to 14 days. If you are not satisfied for ANY REASON (except misuse or improper handling), return in the original shipping container for a full purchase price refund. Applies to hardware only. Sorry, we cannot refund on software. We have confidence in our products and we know you will be satisfied.

WARRANTY

We offer a six months warranty on parts and labor against defects in materials and workmanship. In the event service becomes necessary for any reason, our service department is fast, friendly and cooperative. Our goal is 48 hour turnaround on all warranty or repair drives!

100% TESTED

AEROCOMP disk drives are 100% subjected to burn-in and bench test. We even enclose a copy of the test check list, signed by the test technician, with each drive. AEROCOMP means reliability!

ORDER NOW!

Order by mail or call TOLL FREE TO THE NUMBERS BELOW. Please note toll free lines will accept orders only. We accept VISA or MASTERCARD. Be sure to include card number and expiration date. We will not charge your card until the day we ship. Order by mail with credit card or send check or money order. Personal checks welcome. No delay. Order COD. No deposit required but all COD's will arrive cash, certified check or money order only. We'll send a card showing the exact COD amount before your shipment arrives. Shipping is **not** included in the prices shown. Texas residents add 5% sales tax.

CALL TOLL FREE FOR FAST SERVICE
(800) 824-7888, OPERATOR 24
FOR VISA/MASTERCHARGE/C.O.D. ORDERS
California dial (800) 852-7777, Operator 24. Alaska and Hawaii dial (800) 824-7919, Operator 24.
TOLL FREE LINES WILL ACCEPT ORDERS ONLY!
For information only call (214) 339-8324.
For TECHNICAL ASSISTANCE call (214) 337-4346.

Dealer inquiries invited

AEROCOMP
Redbird Airport, Bldg. 8
P.O. Box 24829
Dallas, TX 75224

✓82

✓ See List of Advertisers on Page 291

The Christmas Pageant

by Richard Ramella

The best Christmas pageant I ever saw was in the third grade. Icky Vannoy played Santa Claus. A serious actor, he insisted on filling his pillow case sack with real toys. He could hardly lift it.

All he had to do was come running out when the kids on stage yelled, "Santa is here!" Out he ran from stage left, and the weight of his sack propelled him straight ahead for a quick exit off stage right. He kept going.

Icky went home and didn't come back to school. Ever. I think he moved to another state during Christmas vacation.

But you're not here to listen to old third grade stories. You want to be in the Fun House Christmas Pageant.

This month I have a secret Yule Greeting program and A Talk with Santa, each with separate Level II and Color listings.

I also have something else. Like Santa, I've been getting letters. Some Fun House visitors don't believe I know how to program in Extended Color Basic. Well, it's not true. I learned how last night and I wrote St. Nick Portrait to prove it.

So far, I've tried to make as many Fun House programs as possible available for as many TRS-80 models as possible. With the introductions of the Model 100 and the Micro Color

```
100 REM * YULE GREETING * TRS-80 LEVEL II BASIC
110 REM * FUN HOUSE * DEC. '83 * RICHARD RAMELLA
120 CLS
130 A=RND(1022)
140 IF A>133 AND A<138 OR A=142 OR A=143 OR A=198 OR A=199 OR A=20
1 OR A=202 OR A=206 OR A=207 OR A=262 OR A=263 GOTO 130
150 IF A=266 OR A=267 OR A=270 OR A=271 OR A>273 AND A<284 OR A=32
6 OR A=327 OR A=331 OR A=332 OR A=334 OR A=335 GOTO 130
160 IF A=338 OR A=339 OR A=346 OR A=347 OR A=390 OR A=391 OR A>395
 AND A<400 OR A=402 OR A=403 OR A=410 OR A=411 GOTO 130
170 IF A>413 AND A<424 OR A=466 OR A=467 OR A=474 OR A=475 OR A=47
8 OR A=479 OR A>529 AND A<540 OR A>541 AND A<548 GOTO 130
180 IF A=554 OR A=555 OR A=606 OR A=607 OR A=618 OR A=619 OR A>669
 AND A<680 OR A=682 OR A=683 OR A=746 OR A=747 OR A>809 AND A<820
GOTO 130
190 PRINT @ A,CHR$(191);
200 GOTO 130
210 END
```

Yule Greeting—Level II.

```
100 REM * YULE GREETING * TRS-80 COLOR BASIC 4K
110 REM * FUN HOUSE * DEC. '83 * RICHARD RAMELLA
120 CLS(0)
123 GOTO 130
125 PRINT @ A,CHR$(143+48);
130 A=RND(510)
140 IF A=67 OR A=68 OR A=72 OR A=99 OR A=101 OR A=104 OR A=131 O
R A=134 OR A=136 OR A>137 AND A<144 GOTO 125
150 IF A=163 OR A=167 OR A=168 OR A=170 OR A=175 OR A=195 OR A=2
00 OR A=202 OR A=207 OR A>208 AND A<215 GOTO 125
160 IF A=234 OR A=239 OR A=241 OR A>265 AND A<272 OR A>272 AND A
<276 OR A=280 OR A=305 OR A=312 GOTO 125
170 IF A>336 AND A<343 OR A=344 OR A=376 OR A>407 AND A<414 GOTO
 125
180 PRINT @ A,CHR$(207);
190 GOTO 130
200 END
```

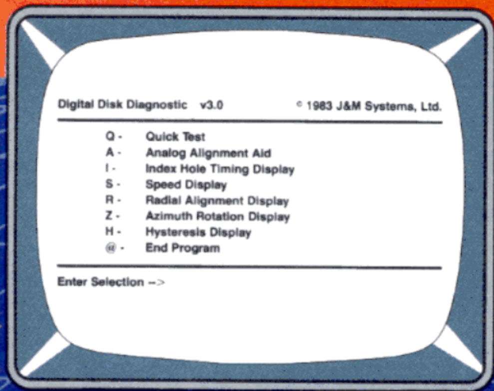
Yule Greeting—Color Basic.

The Key Box

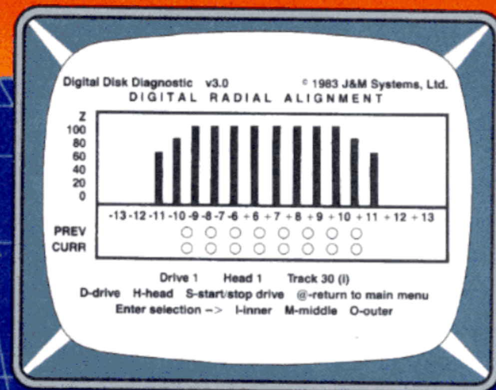
Model I and III
Color Computer
4K RAM
Level II Basic
Color Basic
Extended Basic

DISK DRIVE ANALYSIS PROGRAM

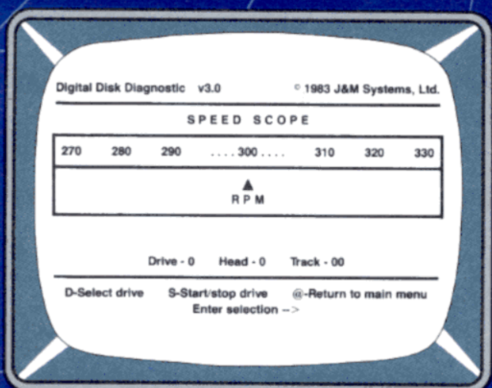
. . . A UNIQUE APPROACH TO DISK RELIABILITY!



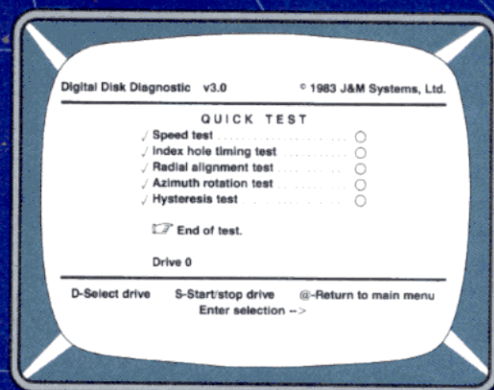
Select any one of seven tests to perform preventive maintenance or to isolate problems. Simple, single-letter commands make DDA easy to use! Use DDA to align the head, adjust the index hole detector, or adjust the speed.



Use the DDA Radial Alignment Test to check the head alignment of your drives. No need for an oscilloscope or other expensive test equipment!



Check the motor speed of your drives. Or, you can even use the Speed Test to adjust the drive speed. No need for any test equipment!



Use the Quick Test to quickly and automatically test five of the most important performance parameters of your drive. Monitor your drives for long term drift. Isolate problems quickly and automatically!

PROTECT YOUR DATA.

Now you can make sure your data is being recorded properly by the use of the revolutionary *Disk Drive Analysis Program (DDA)!*

The *Disk Drive Analysis Program* from J & M Systems, tests your disk's performance and calibration without any additional equipment! It measures your disk's performance and displays it on your screen.

This is the most comprehensive disk diagnostic program available for your TRS-80 microcomputer. You can even adjust drive alignment while watching the display!

Spot problems BEFORE they endanger your data! If you own a disk drive, you NEED the *Disk Drive Analysis Program!*

Now Available For The TRS-80 Microcomputer.

J & M SYSTEMS IS THE DRIVING FORCE!

J & M SYSTEMS, LTD. ✓101

137 UTAH NE · ALBUQUERQUE, N.M. 87108 505/265-1501

Have a merry computer and a happy software.

PRINTERS

STAR MICRONICS GEMINI 10	$CALL
C-ITOH F-10 40 CPS P/S	1250.00
C-ITOH 8510 PROWRITER PARALLEL	399.00
C-ITOH 1550 PARALLEL	699.00
EPSON	CALL
IDS 480 MICROPRISM	489.00

BOOKS

CUSTOM I/O MACHINE LANGUAGE	24.85
HOW TO DO IT ON THE TRS-80	22.95
MICROSOFT BASIC DECODED	24.85
MICROSOFT BASIC FASTER & BETTER	24.85
THE CUSTOM TRS-80	24.85
THE BASIC COOKBOOK	5.85
TRS DOS 2.3 DECODED	22.85
TRS-80 DISK & MYSTERIES	16.85
30 COMPUTER PROGRAMS FOR THE HOMEOWNER IN BASIC	8.95
101 PROJECTS FOR THE Z80	15.50

DISKETTES

SCOTCH 3M

S.S.D.DEN 40 TRK 10, 16 sectors	23.50
D.S.D.DEN 40 TRK 10, 16 sectors	36.50
VERBATIM MD 525-01, 10, 16	26.50
BIB DISK CLEANER only	9.80

DISKETTES STORAGE

5¼" PLASTIC LIBRARY CASE	2.50
FLIP & FILE 5¼"	19.95

TRS CP/M FROM PICKLES & TROUT

MOD II, 16, 12 COMPATIBLE	185.00
LNW MOD II COMPUTER w/CP/M 2.2	$CALL
LNW EXPANSION INTERFACE	345.00
LNW 5/8 DOUBLER w/DOS 3.5	181.00
PERCOM DATA SEPARATOR	27.00

ISOLATORS

ISO-2 6-SOCKET	49.95
DG 115-S 6-SOCKET SWITCHED	30.50

TRS HARDWARE

MODEL III DISK DRIVE KITS Contain: Tandon Disk Drives and Micro Mainframes 4 Drive Controller with Power Supply.

KIT 1: 2 TANDON™100-1 & FD3C	638.00
KIT 2: 2 TANDON™100-2 & FD3C	773.00

MODEL I, III SOFTWARE

DOS PLUS 3.5 MOD I or III	99.00
ELECTRIC WEBSTER	118.50
FLOPPY DOCTOR, MOD III	24.50
L. DOS Ver 5.1 MOD I or III	99.00
MAXI MANAGER Ver B.0. w/UTILITY	118.00
MAXI MANAGER UTILITIES ONLY	44.95
MICROCLINIC MOD I	24.50
OMNITERM, SMARTERM Pkg/MOD I / III	79.95
NEW DOS 80 Ver 2.2	122.50
PROSOFT NEWSCRIPT Rev.7 w/Mail Labels MOD I & III	105.00
THE HOME ACCOUNTANT	59.50
ZORLOFF II	63.50

Alpha Byte COMPUTER PRODUCTS

31304 VIA COLINAS ✓473
WESTLAKE VILLAGE, CA 91362

In Los Angeles: (213) 706-0333

By Modem: (213) 991-1604

CALL OUR MODEM LINE FOR WEEKLY SPECIALS.

Satisfaction Assurance — Your satisfaction is assured by our 30 day guarantee on all hardware products we sell. No refunds after 30 days. All manufacturers' warranties are honored by manufacturers. Defective software will be replaced free during the first 30 days, however, no refunds or exchanges on software. Proof of purchase required. All returns must be authorized in advance.

MODEL 4 OWNERS

CONVERT your MODEL I/III PROGRAMS TO RUN ON MODEL 4

Do you have a lot of time and money invested in Model I/III software, well now you can convert Model I/III BASIC programs to Model 4 with **CONVERTR.**

CONVERTR will eliminate unnecessary spaces and insert all required spaces in your BASIC programs. **CONVERTR** will identify lines which contain keywords not supported by Model 4. **CONVERTR** will identify lines and keywords which the Model 4 handles differently. **CONVERTR** is menu driven and includes an option to list your program and error table on your printer.

CONVERTR comes on a disk and includes an instruction booklet on How to Convert your BASIC programs.

CHECK – MONEY ORDER
COD – VISA – MasterCard

CONVERTR
1 DRIVE SYSTEM - 89.00
2 DRIVE SYSTEM - 79.00

ADEL COMPUTER MART
DEPT 10 BOX 195 ✓356
HARTLY, DE 19953

PHONE 5 pm-9 pm M-F
9-5 Sat (302) 492-8463

THE COMPUTER EXPRESS

32 E. Main Street, Milan, Michigan 48160

SALE THIS MONTH ONLY
TRS-80® MODEL IV 100% R/S
** 1575.00 **

WHY BUY FROM COMPUTER EXPRESS??

- 5 YEARS IN MICRO-MAILORDER BUSINESS
- FULL LINE COMPUTER EQUIPMENT CO.
- 100% GUARANTEED QUALITY PRODUCTS
- SPECIALIST IN COMPUTER SET-UP
- TRADE-IN'S WELCOME
- EXTENDED WARRENTY PLAN
- SERVICE & SUPPORT
- ALL AROUND NICE FOLKS!!

Radio Shack®
DEALER

CORDLESS PHONE
*CONTEST *
(313) 439-2474

✓138

```
100 REM * A TALK WITH SANTA * TRS-80 LEVEL II BASIC
110 REM * FUN HOUSE * DEC. '83 * RICHARD RAMELLA
120 CLS
130 CLEAR 1000
140 N=0
150 FOR A=1 TO 3
160 PRINT "R";
170 FOR B=1 TO RND(30)
180 PRINT "-I";
190 NEXT B
200 PRINT "NG"
210 FOR T=1 TO RND(1000)
220 NEXT T
230 NEXT A
240 PRINT STRING$(60,"*")
250 PRINT "HELLO... THIS IS THE NORTH POLE: LEON THE ELF HERE..."
260 PRINT
270 INPUT "TO WHOM DO YOU WISH TO SPEAK";A$
280 IF LEFT$(A$,5)="SANTA" GOTO 380
290 N=N+1
300 IF N=5 GOTO 350
310 PRINT A$" IS NOT HERE";
320 IF N>1 THEN PRINT " EITHER." ELSE PRINT "."
330 INPUT "ANYONE ELSE YOU MIGHT WANT";A$
340 GOTO 280
350 INPUT "THERE'S A GUY NAMED SANTA CLAUS HERE. WANT TO TALK TO H
IM";A$
360 IF LEFT$(A$,1)="Y" GOTO 380
370 PRINT "THEN YOU MUST HAVE A WRONG NUMBER. TOO BAD... IT'S SO N
EAR"+STRING$(6," ")+"CHRISTMAS...": END
380 CLS
390 PRINT "HANG ON. I'LL GO GET HIM."
400 GOSUB 810
410 CLS
420 PRINT @ 480,"HEY SANTA, IT'S FOR YOU."
430 GOSUB 810
440 CLS
450 PRINT "HE WILL BE RIGHT HERE."
460 GOSUB 810
470 CLS
480 GOSUB 810
490 PRINT "HELLO -- HO-HO-HO -- THIS IS SANTA CLAUS SPEAKING."
500 INPUT "WHO IS CALLING ME";A$
510 CLS
520 PRINT "GOOD TO HEAR FROM YOU, "A$". WHERE DO YOU LIVE?"
530 INPUT B$
540 CLS
550 PRINT "MY MY! CALLING ALL THE WAY FROM "B$"!"
560 PRINT
570 PRINT A$;", HOW OLD ARE YOU?"
580 INPUT Z
590 IF Z>12 THEN PRINT "AND YOU... ER... STILL BELIEVE IN ME. I'M
HAPPY TO HEAR THAT."
600 PRINT "I SUPPOSE YOU WANT TO TELL ME WHAT YOU WANT FOR CHRISTM
AS."
610 PRINT
620 N=1
630 INPUT "WHAT WOULD YOU LIKE";C$(N)
640 D=RND(5)
650 IF D=1 PRINT "HMMM... "; ELSE IF D=2 PRINT "I SEE... " ELSE IF
 D=3 PRINT "SO... "; ELSE IF D=4 PRINT "INTERESTING... "; ELSE PRI
NT "WELL WELL... ";
660 PRINT C$(N)
670 PRINT
680 N=N+1
690 IF N<6 THEN PRINT "AND NOW... ": GOTO 630
700 CLS
710 PRINT "I MUST GET BACK TO WORK NOW, BUT LET ME RECORD YOUR LIS
T."
720 GOSUB 810
730 PRINT
740 FOR X=1 TO N-1
750 PRINT C$(X)
760 NEXT
770 PRINT
780 PRINT "SANTA CLAUS CANNOT EVER PROMISE, BUT I WILL SEE WHAT I
CAN DO.  GOODBYE TO YOU, "A$"."
790 PRINT "AND VERY HAPPY HOLIDAYS!"
800 END
810 FOR T=1 TO 1000
820 NEXT T
830 RETURN
840 END
```

A Talk with Santa—Level II.

Computer (MC-10), I see it's a losing battle.

As *80 Micro* phases out coverage of the Color Computer, I must do the same. And as *80 Micro* invites CoCo owners to make the switch to sister publication *HOT CoCo*, I'm happy to do the same. For *HOT CoCo* I write a column called Elmer's Arcade, and its arcade games are a mix of Extended Color Basic, Color Basic, and Micro Color Basic.

Now let's go to the auditorium for the Christmas pageant.

Yule Greeting

This greeting has separate listings for Level II and Color Basic. What is the Yule Greeting? It's one word, and that's all I'm saying.

The only instructions for this program are to type RUN and tap the enter key. Then you watch. The secret greeting forms slowly.

If you want to trim the tree while you wait, you have plenty of time. If you want to make this into a game, a group of people can watch. The winner is the first to recognize the greeting.

In Level II Basic, the word is black on a white background. In Color Basic, it's red on a white background.

A Talk with Santa

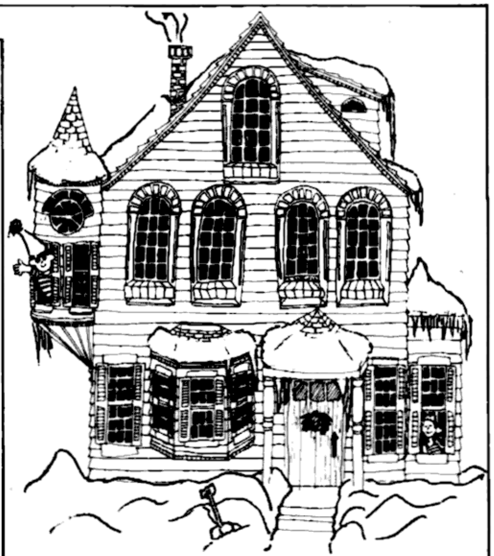
Again, I have separate Level II and Color Basic listings. Look closely at line 100 to make sure you're entering the one for your computer.

In both versions, the program starts with a ringing telephone. In Color Basic you

A Talk with Santa—Color Basic.

```
100 REM * A TALK WITH SANTA * TRS-80 COLOR BASIC 4K
110 REM * FUN HOUSE * DEC. '83 * RICHARD RAMELLA
120 CLS(0)
130 CLEAR 250
140 DIM B(22)
150 DATA 193,3,193,3,193,6,193,3,193,3,193,6,193,3,204,3,176,3,1
85,2,193,10
160 FOR A=1 TO 22
170 READ B(A)
180 NEXT A
190 N=0
200 FOR A=1 TO 3
210 FOR B=1 TO RND(15)+10
220 SOUND 215,1
230 NEXT B
240 FOR T=1 TO RND(1000)
250 NEXT T
260 NEXT A
270 CLS
280 PRINT "HELLO, THIS IS THE NORTH POLE.  LEON THE ELF HERE."
290 PRINT
300 PRINT "TO WHOM DO YOU WISH TO SPEAK?"
310 INPUT A$
320 IF A$="SANTA" OR A$="SANTA CLAUS" THEN 440
330 N=N+1
340 IF N=5 GOTO 400
350 PRINT A$" IS NOT HERE";
360 IF N>1 THEN PRINT " EITHER." ELSE PRINT "."
370 PRINT "ANYONE ELSE YOU MIGHT WANT?"
380 INPUT A$
390 GOTO 320
400 PRINT "THERE'S A GUY NAMED SANTA CLAUS HERE. WANT TO TALK TO
 HIM?"
410 INPUT A$
420 IF LEFT$(A$,1)="Y" GOTO 440
430 PRINT "THEN YOU MUST HAVE A WRONG      NUMBER. TOO BAD... IT
'S SO NEAR CHRISTMAS...": END
440 CLS
450 PRINT "HANG ON. I'LL GO GET HIM."
460 GOSUB 920
470 CLS
480 PRINT @ 480,"HEY SANTA, IT'S FOR YOU!";
490 GOSUB 920
500 CLS
510 PRINT "HE WILL BE RIGHT HERE."
520 GOSUB 920
530 FOR A=1 TO 21 STEP 2
540 SOUND B(A),B(A+1)
550 NEXT A
560 CLS
570 GOSUB 920
580 PRINT "HELLO -- HO-HO-HO -- THIS IS    SANTA CLAUS SPEAKING.
"
590 INPUT "WHO IS CALLING ME";A$
600 CLS
610 PRINT "GOOD TO HEAR FROM YOU,"
620 PRINT A$". WHERE DO YOU LIVE?"
630 INPUT B$
640 CLS
650 PRINT "MY MY! CALLING ALL THE WAY"
660 PRINT "FROM ";B$
670 PRINT
680 PRINT A$", HOW OLD ARE YOU?"
690 INPUT Z
700 IF Z>16 THEN PRINT "AND YOU... ER... STILL BELIEVE  IN ME. I
'M VERY HAPPY TO HEAR   THAT."
710 PRINT "I SUPPOSE YOU WANT TO TELL ME   WHAT YOU WISH FOR CHR
ISTMAS."
720 PRINT
730 N=1
740 INPUT "WHAT WOULD YOU LIKE";C$(N)
750 D=RND(5)
760 IF D=1 THEN PRINT "HMMM "; ELSE IF D=2 THEN PRINT "I SEE...
"; ELSE IF D=3 THEN PRINT "SO... "; ELSE IF D=4 THEN PRINT "INTE
RESTING... "; ELSE IF D=5 THEN PRINT "WELL... ";
770 PRINT C$(N)
780 PRINT
790 N=N+1
800 IF N<6 THEN PRINT "AND NOW... ": GOTO 740
810 CLS
820 PRINT "I MUST GET BACK TO WORK NOW, BUT LET ME RECORD YOUR L
IST."
830 GOSUB 920
```

Listing continued



hear the ring, but in Level II Basic you see the word R-I-I-I-ING! a few times. Then an elf named Leon answers the phone at the North Pole and asks you to whom you wish to speak.

If you don't know who to ask for after spending all that money on a long-distance call to the North Pole at Christmas time, I'm afraid this program ends.

If you reach the right North Pole personality (S***a C***s), you get to tell him your Christmas wish list.

St. Nick Portrait

This part of the big yule show is only for people with Extended Color Basic machines. It draws a picture of Santa Claus. It's a fairly good likeness, although I can't draw hands.

And there's more. You see Santa drawn on the screen and you have a few moments to admire his pudgy form. Then a green text screen appears that asks you to wait a moment.

Behold, on comes a picture of Leon, who had a brief speaking part in A Talk with Santa. In this program he gets a nice closeup but has no lines to speak.

Listing continued

```
840 PRINT
850 FOR X=1 TO N-1
860 PRINT C$(X)
870 NEXT
880 PRINT
890 PRINT "SANTA CANNOT EVER PROMISE, BUT IWILL SEE WHAT I CAN D
O. GOODBYE TO YOU, "A$"."
900 PRINT "AND VERY HAPPY HOLIDAYS!"
910 END
920 FOR T=1 TO 1000
930 NEXT T
940 RETURN
950 END
```

```
100 REM * ST. NICK PORTRAIT * TRS-80 EXTENDED COLOR BASIC
110 REM * FUN HOUSE * DEC. '83 * RICHARD RAMELLA
120 PCLS
130 PMODE1,1
140 SCREEN 2,1
150 CIRCLE(128,96),40,,1.5
160 CIRCLE (141,155),15,,1.5
170 CIRCLE(116,155),15,,1.5
180 CIRCLE(160,60),30,,.4
190 CIRCLE(100,60),30,,.4
200 LINE(110,134)-(135,164),PRESET
210 LINE(135,164)-(135,100),PRESET
220 LINE(80,60)-(166,60),PRESET
230 PAINT(129,96),4,4
240 CIRCLE(129,60),20,3,2
250 PAINT(128,60),1,3
260 FOR Y=20 TO 48
270 LINE(112,Y)-(146,Y),PRESET
280 NEXT Y
290 CIRCLE(128,35),25,3,.8
300 FOR X=112 TO 145 STEP 4
310 LINE(X,40)-(X,60),PRESET
320 NEXT X
330 CIRCLE(128,56),6,6,.5,.0,.5
340 LINE(126,40)-(130,46),PSET,BF
350 LINE(110,22)-(150,27),PSET
360 LINE(160,0)-(110,22),PSET
370 LINE(160,0)-(150,27),PSET
380 PAINT(125,20),4,4
390 CIRCLE(122,34),6,,.6,.5,1
400 FOR X=122 TO 124
410 PSET(X,36,3)
420 PSET(X+11,36,3)
430 NEXT X
440 CIRCLE(135,34),6,,.6,.5,1
450 CIRCLE(72,60),8,3
460 CIRCLE(188,60),8,3
470 PAINT(74,60),5,3
480 PAINT(188,60),5,3
490 FOR Y=108 TO 116
500 CIRCLE(128,Y),40,3,.5,1,.5
510 NEXT Y
520 FOR Y=111 TO 115
530 CIRCLE(128,Y),38,5,.5,1,.5
540 NEXT
550 FOR Y=170 TO 190
560 CIRCLE(114,Y),8,3
570 CIRCLE(142,Y),8,3
580 NEXT
590 FOR X=95 TO 112
600 CIRCLE(X,190),7,3
610 CIRCLE(X+48,Y),7,3
620 NEXT X
630 CIRCLE(163,3),4,3
640 FOR T=1 TO 5000
650 NEXT T
660 CLS
670 PMODE 4,1

680 PRINT "WAIT JUST A MOMENT MORE AND YOU WILL SEE ONE OF SANTA
'S ELVES... LEON."
690 LINE(0,96)-(255,191),PSET,BF
700 FOR T=1 TO 1000
710 NEXT T
720 SCREEN1,1
730 GOTO 730
740 END
```

St. Nick Portrait.

FUN HOUSE

If you want to see only Santa and not Leon, put a new line (655 GOTO 655) into the program.

Next month is 1984. I wonder how many days each of us has been alive. We'll try to figure that out an easy way. Next month's programs will work on any TRS-80 model. ■

Do you have a question about a Fun House program? Write me, Richard Ramella, 1493 Mt. View Ave., Chico, CA 95926. You must include a 20-cent stamped, self-addressed envelope. From other countries, send a self-addressed envelope and your nation's coin equal to postage.

✓ See List of Advertisers on Page 291

FEEDBACK LOOP

by Terry Kepner

Send any questions or problems dealing with any area of TRS-80 microcomputing to Feedback Loop, 80 Micro, *80 Pine St., Peterborough, NH 03458.*

The July 1983 column had three letters concerning problems with the Smith-Corona TP1 and SuperScript. I, too, have a TP1 printer, as well as a DMP-2100.

I tried the DW2 driver and had the same difficulties as the other people. Using your ideas, I located the place in the DW2 driver program where initialization occurs, address BE0F hexadecimal (hex).

Instead of using an editor/assembler to change the code, I used the F function (file patch utility) of Debug on my Model III disk system. The procedure is: Enter Debug at the TRSDOS READY prompt, press the F key, enter DW2/CTL for the file specification, page through the file until line number 3F00 appears on the screen, press the M key, place the cursor over the byte at 3F03 (19), enter 00 and press the enter key, and exit Debug.

I enjoy 80 Micro *very much. I am a relative newcomer to computing and appreciate the hints and advice in your column.*

W.C.
Tampa, FL

Thanks for the modification instructions and your kind comments. For a "relative newcomer" you sound as if you're doing pretty well.

First, I would like to add CP/M to my 48K single-disk Model I. Where can I get the hardware and software to do this, and how hard is it to do?

Second, I want to add a modem to my computer. Since I don't have an RS-232 board, where can I get one? Or should I buy a modem with a built-in RS-232? And could I use the RS-232 for other purposes if I did this? Is there a local CompuServe number in the Champaign/Urbana, IL area?

And finally, what have you heard about the new Banana printer from Leading Edge Products? At $249.95 it sounds like a good buy.

T.D.
Mascoutah, IL

Several dealers sell CP/M upgrades for the Model I computer. For a list of them, see last month's column. All are fairly simple to install.

If you want a Radio Shack RS-232 board for your computer, you can still buy one from Radio Shack. If your local R/S doesn't have any in stock, you can buy one from Rider Radio in Peterborough, NH 03458. The problem with the Radio Shack unit is that it tends to make poor contact with the expansion interface (EI) main circuit board and is prone to glitching while in use. Several weeks ago mine started having problems and ended up getting me thrown off the CompuServe network three times in 15 minutes. The only real solution to the poor contact problem is to solder the RS-232 to the EI connection (not a job for amateurs).

Getting a combination RS-232/modem unit for your computer would make it easier to use and would eliminate most of the contact problems (especially if you use Gold-Plug 80s on the EI extension port). The disadvantage is that the RS-232 ends up being dedicated to the purpose, although at least one manufacturer gives you an extra edge connection for the RS-232.

Your final choice is to buy an external RS-232 board that plugs into the expansion port of the RS-232. This is the most common approach used by people unwilling to put up with the Radio Shack RS-232.

Both of the last two choices have one problem; they don't use the same address ports as the R/S board and this causes some difficulties finding software. If you decide to buy an outside RS-232 board, check with the manufacturer about which software will or won't work with their board.

I've heard nothing about Leading Edge's Banana printer except that it's available. The September *80 Micro* has some information in the New Products section (p. 349).

What jumper changes are required when upgrading a Model III or 4 with 64K to 128K?

T.K.
Julian, CA

You don't need any jumpers. You must replace the DIP (dual inline package) shunt in socket U72 with a Programmable Logic Array (PLA). The ones used by Radio Shack are made by Monolithic Memories Inc., and are called PALs (a registered trademark).

The PLA used with the Model 4 comes with the 128K upgrade kit. I have no information on the serial number or specifications of Radio Shack's PAL.

We just purchased a new Model 4, and I want to know if anyone has written any drivers to use the 80-column screen and 4 MHz clock speed with Model III programs? I currently have DOSPLUS 3.5 for the Model III but am unaware of any drivers or patches to use the Model 4 features.

Second, I'm going to be writing some estimating programs on the Model 4 requiring continuous calculations and comparisons with large indexed numeric data files. I want to know what languages you feel best suit my needs. I'm using Basic, but it's time-consuming and the file handling is too complex.

J.R.F.
Huntington Beach, CA

No one has as yet written a driver to use the Model 4 features in the Model III mode, but it won't be long before the DOS manufacturers have fixed their DOSes to optionally use those features.

I know it's possible because Holmes Engineering has several DOSes patched to use their 80-column board upgrade for the Model III. Once the DOSes are ready, the Basic programs can easily use the features and software will be developed that takes advantage of the Model 4 options in the Model III mode.

Languages are always a problem. Cobol was designed for business, and is used to manipulate large numeric arrays as in inventories and data bases, but its I/O routines are as weak as Basic's. Fortran was designed as a scientific language for extremely fast and accurate numeric calculations. Basic was intended as a good beginner's language, with good string handling capabilities (Cobol and Fortran's weak points), moderate calculation accuracy, and easy-to-program I/O routines (which become complex as they get larger).

If you want a fast calculating language and I/O routines that you can customize and optimize for speed, I suggest you investigate Forth. Forth is a bit unusual in that it is totally flexible. As a Forth programmer you can define new Forth commands to do whatever you want done, much like defining a new Basic command in Basic. For example, you could define a command to clear the screen, load the next record into memory, and display it on the video by assigning a name to the sequence of Forth commands that do the same action.

Forth uses a syntax that's radically different from all other languages, but it's highly efficient. This syntax difference is the most difficult part of learning Forth. A page of Forth commands (defined as one display screen) looks like a scrambled-code message. One common complaint is that a few days or weeks after writing Forth code, not even the programmer can figure out exactly what he did.

Once you learn Forth, you'll love it for its programming ease. If the command isn't there, just define your own new command. In fact, asking a Forth aficionado how to do something (such as writing a custom disk I/O routine) can be frustrating because he'll usually tell you to just write it. Currently, the best Forth available for the TRS-80 computer is from Miller Microcomputer Services (61 Lake Shore Road, Natick, MA 01760, 617-653-6136, $129.95).

I have a Model I with an LNW Expansion Interface and a Teac disk drive. On power-up, the disk comes on for about three seconds and then shuts off. Reset doesn't do anything differently. After the drive shuts off, the keyboard goes dead and I get no response.

I checked all the connectors, solder joints, proper parts in sockets, etc., and everything seems in order. Now what?

K.T.
Salem, NH

I need more information. Does the unit work as a 16K Level II machine? Does the unit work as a Level II machine with the interface attached and turned on? Does the interface have memory? If it does, can you get it in Level II mode (does it power-up as a 32K or 48K machine)? If there's no memory in the interface, are you sure your DOS will operate in only 16K? Some DOSes require a minimum of 32K to work right. What DOS are you using? Have you called or written the LNW trouble desk for help? Is your interface a kit you built, or did you buy it?

If you get full memory from the interface and the computer works fine as a Level II machine, something is wrong with the disk drive controller circuitry or with the disk drive itself. Find someone who has a working Model I disk system and use your keyboard on it. If it works, your keyboard unit is OK. Have your friend use your drive on his system. If it works, the problem is not with the drive. If the keyboard unit and the disk drive test out OK, the trouble is with the interface.

As you know, the PRINT #-1 instruction on the Model III is very slow because it always writes the header first. I would like to know if there's a buffer I can fill before writing to cassette. I know the Color Computer PRINT #-1 works like that.

I also want to upgrade my system to 32K. I bought one 4116 chip and inserted it in my computer as you described in your column. It didn't work; I still get 15314 as the memory available. Does the computer read blocks of 16K, or is the chip bad?

I need a good memory map of my computer, can you recommend a good book?

J.P.
Berchem, Belgium

The design of the cassette I/O routines of the Model III and Color Computers are totally different. The only way you'll get the Model III to buffer cassette I/O in the manner of the Color Computer is to write your own machine-language cassette routines.

You upgrade the memory of all the microcomputers made by Radio Shack (except their pocket computers) in blocks of 16K. The computers address each chip in the 8-chip block as 16K by 1 bit. Installing only one chip activates only 1 bit of the 8-bit words in the 16K block. To upgrade, you must use all eight chips.

Soft Sector Marketing sells the book *Model III ROM Disassembled*, which includes a memory map of the machine.

I read D.J.'s question concerning the installation of new characters in the Model I character generator (August 1983, p. 316). Enclosed with this letter is some literature on a character generator ROM that I manufacture. Note that besides having true lowercase descenders, you can also program the ROM to contain a second character set custom designed by the user.

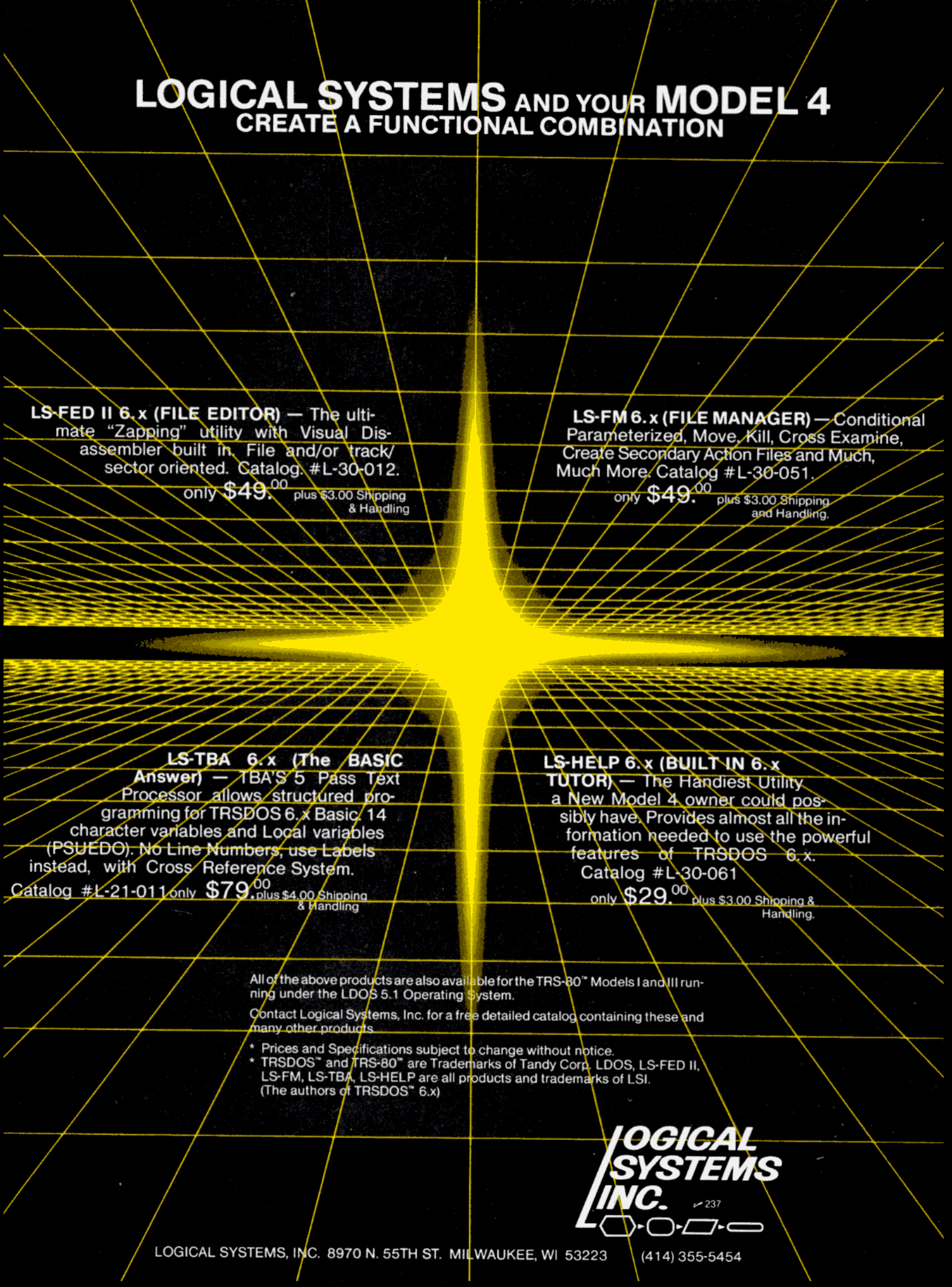
T.W.
Bainbridge Is., WA

Now that's an interesting development. According to the literature, the EC-1 is a pin-for-pin replacement ROM for the MC M6670 used by the Model I as a character generator. In addition to replacing the TRS-80 character set with one containing true descenders, the EC-1 has room for a complete alternate character set. The characters are 5 by 8 matrices, so you can custom design any character you want.

If you want to use the alternate character set and the normal set, you can install a toggle switch to let you pick the one you want. Or you can fix the ROM to use only one of the two sets available. The EC-1 is also inexpensive, retailing at $11. If you want a custom character set, the EC-1 costs $19.

Of course, to use the upper-/lowercase generator, you'll have to have modified the video RAM so that it displays the additional characters. You can do this easily by soldering a 2102 RAM on top of the video RAM chip, as explained in the April 1981 issue of *80 Micro*.

Address your questions to: Electronic


LOGICAL SYSTEMS AND YOUR MODEL 4
CREATE A FUNCTIONAL COMBINATION
LS-FED II 6.x (FILE EDITOR) — The ultimate "Zapping" utility with Visual Disassembler built in. File and/or track/sector oriented. Catalog #L-30-012.
only $49.00 plus $3.00 Shipping & Handling
LS-FM 6.x (FILE MANAGER) — Conditional Parameterized, Move, Kill, Cross Examine, Create Secondary Action Files and Much, Much More. Catalog #L-30-051.
only $49.00 plus $3.00 Shipping and Handling.
LS-TBA 6.x (The BASIC Answer) — TBA'S 5 Pass Text Processor allows structured programming for TRSDOS 6.x Basic. 14 character variables and Local variables (PSUEDO). No Line Numbers, use Labels instead, with Cross Reference System.
Catalog #L-21-011 only $79.00 plus $4.00 Shipping & Handling
LS-HELP 6.x (BUILT IN 6.x TUTOR) — The Handiest Utility a New Model 4 owner could possibly have. Provides almost all the information needed to use the powerful features of TRSDOS 6.x.
Catalog #L-30-061
only $29.00 plus $3.00 Shipping & Handling.
All of the above products are also available for the TRS-80™ Models I and III running under the LDOS 5.1 Operating System.
Contact Logical Systems, Inc. for a free detailed catalog containing these and many other products.
* Prices and Specifications subject to change without notice.
* TRSDOS™ and TRS-80™ are Trademarks of Tandy Corp. LDOS, LS-FED II, LS-FM, LS-TBA, LS-HELP are all products and trademarks of LSI. (The authors of TRSDOS™ 6.x)
LOGICAL SYSTEMS INC.
237
LOGICAL SYSTEMS, INC. 8970 N. 55TH ST. MILWAUKEE, WI 53223 (414) 355-5454

Closet, 8187 Blakely Ct. West, Bainbridge Is., WA 98110.

I ran the MEMTEST program that came with TRSDOS on my computer. The checksums for the A and B ROMs were OK, but the ROM C checksum was different from the ones listed for it. My Model III ROM C has a checksum of 2F84. Is there a reason for concern about the difference? My computer has worked flawlessly since I bought it.

L.L.
Bronx, NY

Worry not, the latest ROM C has the checksum you have. It's just that the program is an older version and doesn't have all the correct checksums for the ROM C versions available. Happy computing.

I'm confused about the sound routines for the Model III computer. I would like some information about the way people work with them. I've tried to put their routines on the end of my programs and all I get is a Cass? prompt.

I tried to modify the Memory Size?, PEEK, and POKE. I also tried four different kinds of sound routines; they all work as long as the routine is by itself. Does it have anything to do with POKE 16526,n and POKE 16527,n?

T.R.W.
Southern Pines, NC

There are several things that could be wrong. First, where are these sound routines going in memory? If they're located at a particular place in memory, moving them around will make them malfunction. If the routines reside at the top of memory, only one can be there at a time without fouling up the others.

If a given routine is located at the top of memory, you have to use the Memory Size? prompt to protect the routine from being stepped on by Basic when it tries to run your program. You also have to tell Basic that there is a machine-language routine located at the top of memory, and that you want to use the routine. This is done with the two POKEs you mention. These place the address of your sound routine in a location reserved by Basic for the USR function, which passes control from a Basic program to a machine-language program and back.

So, if you want to use the sound routines in your programs, you first have to determine where the routines are located. If more than one routine is located there, you have to make sure that the routines don't overlap, or you'll ruin one or all of them. Similarly, cassette-based systems support only one USR function at a time, so if you have several routines, you'll have to develop a method to switch between the routines when you call the USR routine (perhaps use a GOSUB, before the USR command, to jump to lines that POKE the starting address of the particular routine you want into the USR address location).

Start with a routine that you know works; modify the program to do what you want. If it no longer works, you'll have to contact the author for more information on what the routine does and how to modify it. If it does work, start on the next routine.

I have a 48K Model I with three disk drives and a Model 33 teletype for a printer. I have also acquired another Model I, which is a 16K Level I computer. I intend to upgrade it to Level II, and have it as a spare CPU for the first computer.

Can I hook up the two units so that I can control the main unit with the spare? I'd like to have the spare keyboard and monitor upstairs in the livingroom so I can write code, letters, or whatever. I'd also like to be able to load programs into the main unit, alter them, and then save the results on disk. I have an interface that lets me turn the printer on and off without commands.

As I see it, the new keyboard will control the old computer, printer, and drives, and the new monitor will report what's happening. I have no need for the paths to be reversed.

J.C.
Wayne, NJ

In other words, you want to be able to make the older computer a slave to the new one.

The only method I know of for such an application requires the use of RS-232 boards. You load a host terminal program that treats all RS-232 incoming data as if it were typed on the computer's keyboard. All video display is simultaneously echoed out the RS-232. The other computer would have a terminal program loaded in memory that simply sends every keystroke to the RS-232 and takes incoming data and places it on the video display. Nothing is stored or kept in the remote computer's memory, except the program itself.

This method works the way you want. The disadvantage to this system is that you'd have to load the terminal program into the remote machine by cassette tape, and load the host program from disk. Every time you wanted to use the entire system, you'd have to make sure the host computer has the host program in memory and runs correctly before loading and using the remote computer.

If you wanted to get fancy, you could buy BBS software for your disk computer and leave the computer permanently turned on. Whenever you wanted to use the system, just call it up with the remote program (go on-line with the RS-232 terminal program) and start using your computer. Or leave the BBS on drive zero with Auto engaged to load and execute the BBS program. Then you would just have to turn the computer on when you want to use it.

The disadvantage to this system is that you have to buy the additional hardware for the two RS-232 boards, and run an RS-232 cable between the two machines (maximum length without an intermediate amplifier is about 25 feet). For your remote computer, you would have to buy a non-Tandy RS-232 board that lets you attach directly to the keyboard unit without an expansion interface. The disk-based computer would likewise require a non-Tandy board (if you want reliable operation for long periods of time).

I have a strange problem that I hope you can help me with. I have a 48K, dual-disk Model I computer that randomly reboots when in Disk Basic. The computer works fine in DOS, Level II Basic, and when running machine-language programs. I have cleaned all the connectors and resoldered everything that looked flakey. I'm using NEWDOS80, but I'm sure that isn't the problem.

G.S.
Danbury, CT

You didn't mention the age of your computer. If you have one of the older expansion interfaces, you may need to have it modified for increased reliability. The first step is to use the buffered cable; the EI-to-CPU cable has a black box in the middle of it. This helps stabilize the data line and makes it more resistant to electrical noise. If the problem persists, you have to direct-wire the RAS, CAS, and MUX lines inside the keyboard and EI units, from the origination points to the edgecard connections.

If you still have problems, you have to make the pregnant cable (MUX) modification. The three previously mentioned bus lines are cut from the EI–CPU cable and placed in a separate, twisted wire pairs, DIN cable. This cable has a male and female six-pin DIN connector in the middle to let you unplug the CPU unit from the EI unit (hence the bulging, pregnant look to the cable).

These modifications are required on most old model expansion interfaces, which were poorly designed. If you already have these modifications, remove the two power supplies from the EI case; the fluctuations of the magnetic fields induce interference with your EI RAM chips. If you're still having problems, check the FDC chip for poor connection to the circuit board.

I would like to learn Assembly language, but all I know is that it's faster than Basic. I have the book Inside Level II, *but I don't understand it. Please tell me the best way I can learn about Model I Level II Assembly language, without having to spend lots of money.*

T.K.K.
Chico, CA

First, the Model I computer is a Z80 CPU-based computer, so any book teaching the theory of Z80 programming will help you. However, I suggest that you get either *TRS-80 Assembly-Language Programming* (Radio Shack #62-2006, $3.95) by William Barden, or *TRS-80 Assembly Language* (Prentice-Hall, Englewood Cliffs, NJ 07632, $9.95) by Hubert S. Howe Jr. Both books deal specifically with the Model I. Howe's book is easier to understand, while Barden's book gives instructions on using Radio Shack's T-Bug and Editor/Assembler.

A year ago, I purchased a Microtek Bytewriter-1 and their interface board for my Model I. Everything worked fine until I added the chips for 48K operation. Level II recognizes the full 48K when first powered up, but after a variable time the computer reboots to the Memory Size? prompt, or some other apparently random spot.

I thought the problem was interference until I discovered that responding to Memory Size? with 32767 or less prevents the problem while still allowing machine code to run in high RAM. What's up?

K.H.
Livermore, CA

Basic stores program variables, arrays, and program operation tables starting at the top of RAM and working down. You use the top 2 bytes constantly. Since the dividing address between the keyboard unit and the expansion memory is 32767, your difficulty is with the high 32K RAM bank.

First, make sure that the connection between the keyboard unit and the expansion box is clean and solid (use isopropyl alcohol and cotton swabs to clean off the dirt, and a pink rubber eraser to remove any tarnish off of the cable contact; the connectors should fit tightly).

If the problem persists, try swapping the top 16K bank with the memory in the keyboard. Next, check to see if the R/S black box power supplies are too close to the memory or the interconnecting cable. And finally, is the RAM and printer circuitry being driven by the keyboard's power supply? If so, the interconnecting cable could be interfering with itself, although that's not too likely. What's more likely, if the expansion unit uses the keyboard power supply, is that the added memory is overtaxing the power supply and causing your trouble. In this case you'll have to get a separate power supply for the expansion box.

If the problem still doesn't go away, contact the manufacturer of your expansion box and ask for help.

I have a 48K, three-drive Model I with the Radio Shack doubler installed. My problem is with TRSDOS 2.7DD. When attempting to copy from single-density to double-density, TRSDOS 2.7DD won't read anything above track 35 on the single-density disk.

It appears that this problem could be corrected by a very simple patch, but Fort Worth won't say when or if the patch will be available, nor will they release the file specification, password, sector location, or any other data on how to access TRSDOS 2.7DD.

Do you know of a patch for this problem, or must I start looking for a new double-density DOS?

R.K.
Clovis, NM

You'd better start looking, because I don't have that patch in my bag of tricks. Can anyone else help?

My 48K, single-disk Model I has several problems: spontaneous disk reboots, inability to reboot a disk using the keyboard reset button, DOS message CANNOT BOOT or something similar while the disk spins in the drive, and occasional screwy behavior while using VisiCalc (i.e., the left arrow key moves the cursor down one line).

Just recently I had to replace the keyboard-CPU cable to get the computer to function as a 16K computer.

I can feel a slight wiggling of the connector at the EI—perhaps the original Radio Shack plug is getting old (I tried replacing the keyboard-CPU cable, but got no improvement). Right now, I haven't had a reboot since I started this letter. Apparently if I get the cable connectors just right, proper contact is made and no rebooting occurs.

I'm thinking of soldering the EI and CPU together with a 40-conductor cable, but feel I might regret it later on. Now what should I do?

S.D.
East Lyme, CT

Things are never simple. First, the keyboard-CPU cable is a vary weak point in the system. When you get strange results after typing on the keyboard, this cable connection is almost always the problem. This cable can also cause spontaneous system reboots, and prevent proper disk-booting action.

✓ *See List of Advertisers on Page 291*

Check it carefully for lifted traces on the edgecard, cold solder joints, or intermittent solder bridges.

Next, check the power connectors where your power supplies attach to the circuit boards; they may be loose and causing reboots. Also, check and clean the contacts on the FDC chip in the EI (that's right, clean the FDC chip pins).

The disk booting problem has several possible sources: The edgecard connection to the disk drives may need cleaning, the disk drive logic card's edge connector may need cleaning, the head may need a slight realignment and cleaning, or the motor speed may need adjusting.

Start with the easiest: clean the edgecard connectors, then use a disk drive cleaner for the head (or alcohol and long cotton swabs).

The reset button sounds as if it's broken. As a Level II machine, does Reset work? It may also be suffering from cold solder joints, lifted traces, and other mechanical problems. Let me know if you're successful in correcting the problem.

I have a problem with my Model I speed-up kit. My system comprises: Model I (board #1700069D, Satellite ROMs, 16K RAM from Exatron, Cerdat's The Patch, Programma 80-Grafix board, and Exatron Speedup Kit); Radio Shack Expansion Interface (board #1700077D, 32K RAM from Exatron, Percom Doubler II); Aerocomp's 80-track disk drive (MPI 91); Exatron Stringy Floppy; Epson MX-80 printer; and DBLDOS from Percom.

I bought the keyboard in 1978 as a Level I machine. When Level II was released, I ordered it the next day. The same goes for the expansion interface, which works fine without a buffered cable or other reliability modifications.

Everything worked fine until I increased memory to 32K (the speed-up kit supports 50 and 100 percent speed increases, both of which function fine in a 16K system). When I try to use the system with 48K, the 50 percent mode works for everything except machine-language programs; the 100 percent mode locks up everything.

Is there a way to have the expansion interface support the high speed operation? Exatron suggested replacing the CPU with a Z80A, and said I might have slow memory. I removed the memory from the expansion interface and tried it in the keyboard. Everything worked fine, so slow memory can't be the problem.

I borrowed a copy of Tandy's Dynamic Memory Test program from a friend and ran it at normal speed with no problems. At the 50 percent speed increase, everything checked out until the 16K/32K boundary, when the system crashed. The 100 percent speed increase immediately locked up the computer.

P.L.
Lawrence, KS

Yes, you can make the expansion interface operate at high speed. You have several points where problems could foul up your high-speed modification. First is the memory. The expansion interface is much more critical of timing constraints than the keyboard unit. Remember, the expansion interface is farther away from the CPU than the keyboard RAM.

Electrons in metal move at about 3×10^{10} centimeters per second. There are 1×10^{9} nanoseconds (ns) in a second. So in one nanosecond, the electrons move about 30 centimeters (almost 12 inches). This may not seem like much, but if you're using RAM rated at 250 ns (time required to respond to a signal from the address bus and put the addressed byte's data on the bus) in the EI, the added distance may be just enough to cause problems.

In other words, RAM that performs flawlessly in the keyboard can fail in the expansion interface. You can see the same problem occur at normal speed by using 400 ns memory in the EI instead of the standard Radio Shack 300 ns memory.

Furthermore, at 3.54 MHz, each clock pulse defines a 282 ns window. If you slow down the signal too much by the address-decoding integrated circuits (ICs) and support chips, the RAM won't get the information on the data bus until it's too late.

The best solution is to use the fastest memory possible; 150 ns is great, 100 ns is the best (but very hard to find, and expensive). That takes care of memory.

Remember that the Model I was an experiment, designed for the tinkerer. It wasn't designed for clock speeds of over 1.77 MHz. This means that the digital components weren't designed for faster speeds either. That is, the units can't respond to the orders from the CPU fast enough to deliver or process the required information within the amount of time allocated by the clock frequency. Result: system lockups.

Some of the components are capable of operating at 5 MHz, but many begin to fall behind the CPU at that speed. Even some Z80 CPUs can't operate at the higher speed and have to be replaced. And for the EI, the greater distance to the EI, plus the additional memory-decoding ICs, can make the difference between high speed operation or a locked up computer.

Also, you say you don't have any of the EI reliability modifications. Well, Radio Shack didn't come up with those modifications for the fun of it. They were developed as a solution to the common complaint of Model I owners about EI memory problems. The reason you've never had to make the modifications is that you didn't have the memory in the EI to cause the problems. All the modifications were made to correct memory problems. If I were you, I would seriously consider having those modifications made.

If you want better explanations about speed-up problems, and instructions on what you can do to correct them, call Holmes Engineering. ■

Terry Kepner is a freelance writer and programmer, and the vice president of Interpro. He's been writing about microcomputers since 1979.

Frequently Needed Numbers

Radio Shack, National Parts Division, 900 East Northside Drive, Fort Worth, TX 76102, 817-870-5662, M/C and Visa accepted, each order has $1.50 handling charge.

IJG Inc., 1260 West Foothill Blvd., Upland, CA 91786, 714-946-5805. Publisher of *TRS-80 Disk and Other Mysteries* ($22.95), *Microsoft Basic Decoded and Other Mysteries* ($29.95), *The Custom TRS-80 and Other Mysteries* ($29.95), *Basic Faster and Better* ($29.95), *Machine-language Disk I/O and Other Mysteries* ($29.95), *TRSDOS 2.3 Decoded and Other Mysteries (Model I)* ($29.95), *How to do it on the TRS-80* ($29.95), and the *Electric Pencil Word Processor* ($89.95).

See our other ads on pgs. 127, 203

80 LIST OF ADVERTISERS

For further information from our advertisers, please use the Reader Service card.

by Amee Eisenberg

The heartbreak of CRC errors

With every turn of its drives, your computer's floppy disk controller (FDC) is on the lookout for errors. The Cyclic Redundancy Check (CRC) safeguards the accuracy of your data by performing a quick calculation on a full sector of data. The CRC byte, the expected solution to the CRC calculation, is just one kind of information encoded on your disk by the disk drive.

To help you better understand how this works, I'll explain disk drive and disk technology. Demystifying the disk operating system's cosmos may make your computing life a little happier.

The Disk Drive

A disk drive does three things: it spins a disk, positions a read/write mechanism, and communicates with the floppy disk controller (FDC) chip. When these processes happen correctly, your computer reads from or writes to your disk. It's something we take for granted until an error message appears on the screen.

Your stomach turns, you pound the reset button praying that it isn't so, and you begin to wonder just what HIT the table or where that sector could have gone. The real question is just what is the drive doing wrong?

Spinning the disk is technology with which everyone who owns a record player is familiar—a motor turns a platter on which you place a disk. Your disk drive spins at 300 rotations per minute (rpm), a speed that could make The Beatles sound like mosquitos. Just as a speed variation on your turntable makes your records sound bad, a speed variation on your drive means bad disk input/output (I/O).

While most manuals suggest that a speed deviation of plus or minus 5 percent is permissible, I get nervous at anything greater than plus or minus 1.5 percent. By the time you've got a 5 percent deviation, you'll also notice plenty of disk I/O problems. Many programs are available that measure the drive's spin speed. Owning one is vital for diagnostic purposes, although actually adjusting the speed requires the derring-do to try open heart surgery on your computer (thus voiding your warranty) and infinite patience. I suggest you let a professional do it.

A stepper motor (stepper motors move discrete distances one step at a time) positions the read/write head. Locating and relocating the tracks on your disk depends on the accuracy with which the stepper motor moves the read/write head. Like the tone arm on your turntable, the stepper motor can position the read/write head to 'play' any track; unlike a record player, the read/write head, as the name implies, also records data onto the disk.

If your disk I/O problems are unpredictable, that is they come and go, watch for a pattern. Does the computer read and write fine when you turn it on but become undependable after it's been working awhile? Suspect your head alignment. A head that's marginally aligned may read fine when you first turn on the computer, but as the machine heats up, the alignment shifts. Again, the easiest solution is taking it to a professional.

Information exists on a disk in the form of magnetic pulses. The head reads more magnetically positive bits as logical ones, the less positive bits as logical zeros. The read/write head picks up these bits in a series, that is, one after another. The Z80 microprocessor at the heart of your TRS-80 computer reads information 8 bits at a time, a method known as parallel. One of the FDC's jobs is collecting the serial input and sending it to the Z80 in parallel.

The FDC navigates the disk by first using the index hole and then using markers written on the disk by the disk operating system (DOS). The FDC looks for two kinds of markers, sector headers and clock bits. Both are written when you format a disk.

TRSDOS Model I				
5	sectors	=	1	gran
2	grans	=	1	track
10	sectors	=	1	track
TRSDOS Model III				
3	sectors	=	1	gran
6	grans	=	1	track
18	sectors	=	1	track
DOSPLUS Model III				
6	sectors	=	1	gran
3	grans	=	1	track
18	sectors	=	1	track
LDOS Model III				
6	sectors	=	1	gran
3	grans	=	1	track
18	sectors	=	1	track

Table. Some examples of different track allocations.

The Disk

When you look at a disk, a few things are apparent. First, a floppy disk is made of two parts, one sandwiched inside the other. The inside is a shiny, metal-oxide-coated, circular piece of mylar plastic (the disk) that slides freely inside the square piece of plastic (the shell) that envelops it. Six asymetrical holes are cut into the disk's shell.

The large hole in the center gives the disk drive a place to grab the disk. To spin the disk in its shell, stick two fingers through the center hole, spread them to apply pressure to the inside rim of the disk, and turn. Do not touch the disk's surface! A small hole in the disk will appear in the round window to the right of the center hole. This is the index hole, which the FDC uses as the absolute landmark of its location on the disk.

The two small half circles punched in the bottom of the disk are alignment notches. They help insure that the disk is set securely in the drive. The square notch cut into the side of the disk is the write-protect notch. If this notch is filled in (by taping a lable over it), the FDC will not let the read/write head write on that disk. To get your computer to write on a write-protected disk, remove the label.

The oval window cut at the bottom of the disk's shell is the place where your drive reads the disk. When you see how small this head access window is, you appreciate how finely tuned the disk drive is. It adjusts the read/write head to find between 35–40 (and on some drives 80) separate positions (tracks) in an area less than 1½ inches long!

The disks you're familiar with are soft-sectored, which means that the

space on their surfaces is partitioned by the DOS software, rather than through a "hard," or permanent, manufacturing process. Because the sectors aren't created in the manufacturing process, DOS designers can divide the disk space arbitrarily.

The DOS

The smallest unit of information a DOS reads or writes is called a sector. Sectors are clumped together as granules or grans, and grans are gathered into tracks. Different DOSes use different patterns of space allocation; some divide a track into two grans with 10 sectors per gran, another uses three grans of six sectors per track. This is why some DOSes can't read disks written by other DOSes. (See the Table for some examples of common DOS allocations.)

Information contained in a sector is of two types, the information you write (data) and the information the DOS writes (sector header). System information includes an identifying address mark (a sort of flag saying "Here's some location information"), the track number, the sector number, the sector length, and the CRC bytes. Combined, this information creates a heading for your data that lets the computer retrieve it when you need it.

In addition to these sector headers, the DOS writes another kind of marker on the disk. Called clock bits because they're written by the FDC's 1 MHz clock, they create spaces between them that hold data. They are, in a sense, the carton; your pieces of data are the eggs.

Using the index hole on the disk, the FDC finds the beginning of a track. Using the clock bits, the FDC locates the spaces where the data resides. When it reads an I.D. address mark (the number FE hexadecimal coming after a gap), the FDC knows it has found a sector. Reading the sector header tells it what track and sector it has found. A lot of space on your disk is taken up just defining and identifying the space you wish to use.

Every time it reads a sector, the FDC comes across 2 bytes of information that are directly related to the accuracy of its last write. These are the CRC bytes. Their value, as mentioned earlier, is determined by a calculation performed on the value of all the other sector bytes. When the FDC reads a sector, it performs this calculation on the data it has just read.

If the value matches the value of the CRC byte on the disk, all's well. If the values don't match, the computer prints an error message on the screen and quits. Depending on the software controlling the disk I/O, the error message says either "CRC error" or "parity error." Either way, it's bad news.

Utilities are available that might straighten out a problem disk: Debug on TRSDOS, SuperZap on NEWDOS80, and Super Utility Plus, to name a few. But honestly, the best medicine for any disk problem is preventive.

Preventive Medicine for Your Disks

Make a back-up copy of every important disk! At *80 Micro*, we use only third-generation copies of our DOSes. The original sat in the disk drive long enough to make a copy of itself, our working master. The working master comes out of its envelope only to create a back-up, the copy we actually use. This policy sounds neurotic—until somebody does one of those ten thousand things that trash disks. But, when that happens, we don't get stuck sitting around waiting for the manufacturer to replace our disk.

The same holds true for data disks; unless you love typing, keep up-to-date back-ups on your data files. It takes time, but ten minutes a week now may save you hours trying to reconstruct a blown disk later.

The rule to remember is, if it doesn't have a back-up, it will develop an error. Back up your disks!

The second great preventive measure is to keep your drives well maintained. Invest in a diagnostic disk and test your drives once a month. If the disk speed varies more than 5 rpm in either direction (faster or slower), get it fixed before it causes problems.

Buy a disk drive cleaning kit and use it. Dirt on your read/write head acts like sandpaper on your disk, an undesirable condition.

Finally, a reminder of stuff you already know. Open and close your disk drive's door gently. If you slam it around day in and day out, it will break. Since the door is part of the mechanism that holds your disk in place, a loose or crooked door will result in disk I/O errors. In a worst-case scenario, a door off its hinges renders the drive inoperable.

Store your disks in their envelopes, away from dust and magnets. Remember, the telephone and the speaker in your cassette recorder have magnets. Keep your disks away from them. The top of your video display screen is also a bad place for disks. It generates enough radio frequency interference to garble your disks.

Your disk drive, disks, and DOS work together in a complex system designed to guard against errors, but, as with any sophisticated technology, it requires care on your part to keep it working. ■

Article	Page	Cassette File Spec	Disk File Spec	Comments
Side A				
		A	TITLE/BAS	Basic
Channels of Communication	140	B	SATCOM/BAS	Basic
Chameleon Code	170	C	CONVERT/BAS	Basic
Assembly-Language Disk I/O	88	DISKIO/CMD	DISKIO/CMD	System
Real World Control—Part II	152	†	CMDTBL/SRC	EDTASM*
Real World Control—Part II	152	†	ALARM/SRC	EDTASM
Real World Control—Part II	152	†	CLOCK/SRC	EDTASM
Side B				
Real World Control—Part II	152	CMDTBL/CMD	CMDTBL/CMD	System
Real World Control—Part II	152	SPRINK/CMD	SPRINK/CMD	System
Real World Control—Part II	152	ALARM/CMD	ALARM/CMD	System
Real World Control—Part II	152	CLOCK/CMD	CLOCK/CMD	System
Real World Control—Part II	152	†	SPRINK/SRC	EDTASM

†Source code not available on cassette.

*EDTASM is a trademark of Radio Shack.

December 1983 Load 80 directory

TRS-80 MODEL 16

XENIX

- CONSULTING
- SYSTEM CONFIGURATION
- INSTRUCTION

INTERACTIVE SYSTEMS DESIGN
SUITE 201, 170 BROADWAY
NEW YORK, N.Y. 10038
(212)-233-6923 ✓147

TEACHERS!

STUDENT GRADE SYSTEM

The complete grading package for the TRS-80 Model III or 4.
16K tape or 32K disk versions.
Maintain multiple class lists.
Use number or letter grades.
Weighted test scores allowed.
Make changes fast and easily.
Print or display class lists, scores and grade averages.
65 page manual, sample session.
Automatic data protection.
Price: $75.00 ✓83

Orders and inquiries, write:

Tinker Techniques
435 Greenway Ave.
Trenton, NJ 08618

TRSDOS↔CP/M

with

REFORMATTER™

Model II users! Convert files between TRSDOS and CP/M!

- **REFORMATTER** runs under TRSDOS
- Operates on single drive system
- Converts in both directions
- CP/M operating system not needed
- All TRSDOS record lengths supported
- Initializes blank CP/M diskette
- Displays or dumps CP/M files
- Manipulates CP/M directory under TRSDOS ✓380

$249.00 from stock. CP/M↔IBM, TRSDOS↔DEC, and CP/M↔DEC versions of **REFORMATTER** also available at $249.00 from MicroTech Exports, Inc., 467 Hamilton Ave., Palo Alto, CA 94301 ☐ Tel: 415/324-9114 ☐ TWX: 910-370-7457 MUH-ALTOS

PRESERVE 80 MICRO WITH BINDERS & FILE CASES.

Keep your issues of *80 Micro* handy and protected in handsome and durable library file boxes or binders. Both styles are bound in dark green leatherette with the magazine logo stamped in gold.

File boxes: each file box holds 12 issues, with spines visible for easy reference.
$5.95 each, 3 for $17.00, 6 for $30.00

Binders: each binder holds 12 issues and opens flat for easy reading.
$7.50 each, 3 for $21.75, 6 for $42.00

(USA postage paid. Foreign orders must include $2.50 per item.)

Please state years desired (1980 to 1984).

Send check or money order to:
Jesse Jones Box Corp., P.O. Box 5120, Philadelphia, PA 19141; please allow 6 to 8 weeks for delivery. Sorry, no C.O.D. or phone orders.

UPGRADE Your Model 4 from 64K to

128K For Only $89.95

Kit includes 8 - 64K RAM chips & **REQUIRED** PAL device. You must already have 64K installed or order separately below.

64K Upgrade For Only $69.95

(Kit includes 8 - 64K RAM chips & **REQUIRED** jumper.)

16K For Model 3 $15.95

Kit includes 8 - 16K RAM chips
Rated @ 150 nanosecond access times.

Logix

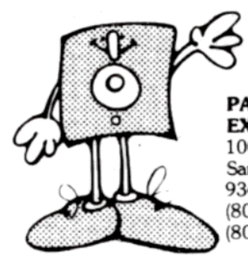
P.O. Box 18982
AUSTIN, TX 78760
512/445-6427

Add $2.00 for shipping. Texas Residents add 5% Sales Tax. No C.O.D.'s, certified check or Money orders for quickest delivery. Personal checks subject to clearance. ✓385

Verbatim

flexible disks

Call Free **(800) 235-4137** for prices and information. Dealer inquiries invited. C.O.D. and charge cards accepted.



VISA

PACIFIC EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA 93401. In Cal. call (800) 592-5935 or (805) 543-1037. ✓207

80micro

Single Back Issues

January 1980 to June 1980 . . . . **3.00**
Single back issue
July 1980 to May 1983 . . . . **3.50**
Single back issue
June 1983 on **4.50**
Add $1.00 per magazine for shipping.

10 or more back issues
add $7.50 per order for shipping. ®

Back issues•Attn. Mail Order
80 Pine St.•Peterborough, NH 03458

TRS-80* SOFTWARE

Write for our incredible catalogue of over 100 pages, containing hundreds of programs for the Model I and III, Colour computer programs available in the Fall. Catalogue is $5.00 refundable on your first order. All software is unconditionally guaranteed for life!! Canadian manufacturer and distributor for Molimerx Ltd. of England.

DEALERS WELCOME

GAMES/DOS/UTILITIES/ BUSINESS PROGRAMS

*LDOS	$159.95	*POWERMAIL PLUS	$199.95
*MULTIDOS	$129.95	*AIRBUS	$ 41.95
*ACCEL 4	$139.95	*DATA WRITER	$189.95
*FROGII	$ 23.95	*DRIVER COMPILER	$ 43.95
*JUMBO	$ 40.95	*ENIGMA	$ 36.95

FULLY SUPPORTED COMPUTER SOFTWARE
*Trademark of Tandy Corporation

JSOFT ✓177

P.O. Box 1437, WINNIPEG, Manitoba R3C 2Z4
(204) 942-0963 CANADA

Trend Software Package

1. Bar Charts
2. Linear Regression
3. Statistics
4. Histograms

$99.95

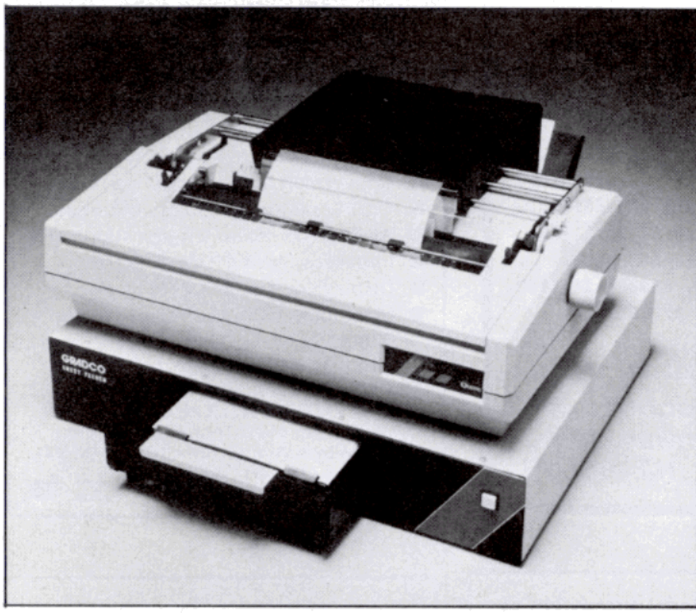
(Add $1.50 for S & H)
(PA Residents Add 6% Sales Tax)
Make Checks or Money Orders Payable To:

Selective Software Systems

SSS
4757 Liberty Avenue
Pittsburgh, PA 15224 ✓53

NEW PRODUCTS

edited by S.F. Tomajczyk



The Over-The-Top Sheet Feeder from Gradco Systems Inc.

Cassette Sheet Feeder

A unique over-the-top sheet feeder for daisy wheel and matrix line computer printers is being sold by Gradco Systems Inc., 3421 West Segerstrom Ave., Santa Ana, CA 92704, 714-549-9175.

The unit features a single-cassette, 500-sheet capacity feeder. To operate, place the loaded cassette under the printer, and when printing begins, paper automatically feeds over the top of the printer directly into the printer's platen. The mechanism is completely enclosed, with no frames or holders located above the printer's profile to obstruct your view.

The feeder accommodates a wide range of paper sizes and envelopes, as well as business forms, and automatic stacks of printed material. You can easily change from sheet to fan-fold feeding. An indicator light and button located at the front of the unit provide auto-stop and jam-release functions.

You can install this over-the-top sheet feeder on many printers, including those from C. Itoh, Diablo, Qume, NEC, and Brother, among others. Pricing ranges from $600–$700 depending on your printer model.

Reader Service ✓ 551

Hamming Around

All you ham operators out there can now receive and transmit Morse code on your Model I, III, or 4 with this software-only Morse Code Interface. This 12K machine-language program has special routines that check incoming signals to make sure that they are valid code and not noise. Bursts of noise are ignored and only valid code is processed and displayed on the video screen.

Another routine lets you view the incoming signal to aid in optimizing the receiver control settings. The program samples the received code bits and automatically adjusts them to the proper speed.

The transmitter mode features five programmable buffers (200 characters total), a type-ahead (working buffer), and user-selectable sending speeds up to 70 words per minute.

Comp-Code 1.0 costs just $26.95 from Gary Woodall Software, P.O. Box 284, Plainfield, IN 46168, 317-271-2565. Dit-dah-dit-dit!

Reader Service ✓ 579

Project Planning Program

Project managers can attest to the work and time involved in getting a project onto the drawing board and then setting it in motion. Now you can reduce those headaches with Plantrac, a comprehensive, menu-driven planning and control system.

It guides you through network creation and amendment (up to 12,500 activities), time resource and cost analyses, scheduling, updating, and reporting (including a user-designed report format). Plantrac generates barcharts, I/J, and Precedence networks on your screen, printer, or plotter with critical path activities highlighted.

A key feature of this program is the resource scheduling (levelling and limiting) component that allows different schedules based on resource requirements and availability. In addition, you can generate progress out of sequence that can be undone if necessary.

Plantrac operates under TRSDOS 2.0 on the Models II, 12, and 16. It costs $3,000 from Computerline Limited, 755 Southern Artery, Quincy, MA 02169, 617-773-0001.

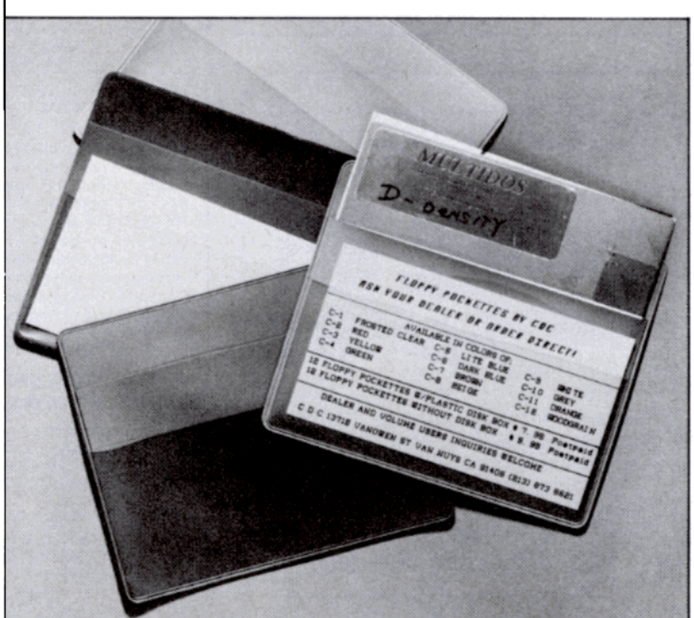
Reader Service ✓ 562

Floppy Pockettes

If you've been stuffing your notes concerning a particular disk in an envelope, or have written cryptic notes on the disk label only to have them smear the next day, Floppy Pockettes may be of some use to you. With these special vinyl disk envelopes, you can store a 3- by 5-inch card in the clear pocket, giving you plenty of room for notes.

The 12 envelopes come in a wide variety of colors for color coding your disks: red, yellow, green, light and dark blue, brown, beige, frosted clear, white, grey, orange, and woodgrain. They fit inside a hinged smoke-plastic box for additional protection.

The 12 Floppy Pockettes with the disk box sell for $7.95; the 12 Floppy Pock-



Keep your notes and disks together with floppy pockettes.

DOES STRING COMPRESSION HAVE YOU TIED UP IN KNOTS?

LET TRASHMAN™ CLEAN UP THE MESS!

THIS PROGRAM IS A MUST FOR EVERYONE WHO USES "BASIC" ON A TRS-80. Why? Because it can reduce BASIC's string compression time delays by 95% or more.

WHAT'S STRING COMPRESSION?

When a BASIC program changes a string (words, names, descriptions), it moves it to a new place in memory, and leaves a hole in the old place. Eventually, all available memory gets used up and BASIC has to push the strings together to free up some space. This takes time. Lots of time. The computer stops running for seconds or minutes, and you may even think it's "crashed".

Yes! String compression is what's been causing all those intolerable delays. The keyboard won't work, and until all the strings have been collected, you just have to sit and wait. Then things run for a while, until string compression is needed again. And again.

If you're using your computer for business, that wastes your money. If you're using it personally, it wastes your time.

WHAT'S THE SOLUTION?

As soon as you start using TRASHMAN, those delays will almost disappear. The program is **very easy to use,** so you don't have to be a computer programmer to take advantage of it. It's written in "machine language" and uses only 578 bytes of memory for itself, plus two bytes for each "string" in your program. It works with other machine language programs and all the major operating systems.

# STRINGS	SECONDS DELAY NORMAL	SECONDS DELAY TRASHMAN	PERCENT IMPROVEMENT
250	11.8	0.7	94
500	45.8	1.6	96.5
1000	179.6	3.5	98
2000	713.2	7.8	98.9

(All timings done on TRS Model I. Model III 15% faster, but pct improvements identical. Listing of timing program available on request.)

HOW WELL DOES IT WORK?

If you use it with a BASIC program that has only a few strings, very little time is wasted in string compression, and TRASHMAN will be only slightly helpful. But, in programs that use hundreds or thousands of strings, including large string arrays, TRASHMAN is just what you need. If you have any remaining doubts, just look at the chart, and then get yourself a copy as fast as possible.

TRASHMAN is available on disk for just $39.95.

ATTENTION SOFTWARE PUBLISHERS:
Trashman may be licensed for use with your packages.
Call for details.

SAVE TIME WITH FASTER

"FASTER" speeds up most TRS-80 BASIC programs by 20-50%. It's helped hundreds of satisfied people and it can help you. Detailed instructions make it easy to use. FASTER analyses your BASIC programs *while they run,* then displays a simple change, usually one line, that sequences program variables so the ROM will find them faster.

You can use FASTER to speed up programs you've bought, as well as programs of your own. Since it isn't a compiler, your BASIC programs can be read and changed afterwards. FASTER works on business programs, models, and games. The more complex your program, the better the results.

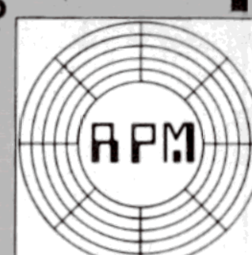
Does FASTER really work? Yes! Just check the reviews in *Personal Computing,* May, 1981, p. 116: **"FASTER is effective and easy to use"**; *80 U.S. Journal,* April, 1982, p. 106: **"I recommend FASTER to everyone"**; and *80 MICRO* (April, 1982, p. 40): **"If you...would like a significant increase in the run-time speed, then buy FASTER."**

FASTER runs on the TRS-80 Models I and III, 16-48K tape or disk, and all major operating systems. **$29.95**

"QUICK COMPRESS" takes only 276 bytes of memory, and removes the blanks and remarks from even the largest BASIC program in less than 3 seconds. It produces smaller, faster programs without altering their logic. **$19.95**

SPECIAL: FASTER and QUICK COMPRESS: **$39.95**

ERRATIC DISK DRIVES?



You can avoid unnecessary disk errors and repair bills by using **RPM.** This easy-to-use program measures the rotational speed and fluctuations of your disk drives, and warns you if they are running **too fast, too slow, or unevenly.**

Incorrect or erratic speed is a common cause of unexplained disk errors and loss of data. RPM's documentation explains how to **detect** *and* **correct** these problems quickly and easily. As *80 MICRO* (April, 1982, page 41) said: **"If your drives have problems I recommend RPM before paying to get it repaired."**

RPM is supplied on diskette for the TRS-80 Models I and III. We suggest you order a copy *before* you need it.

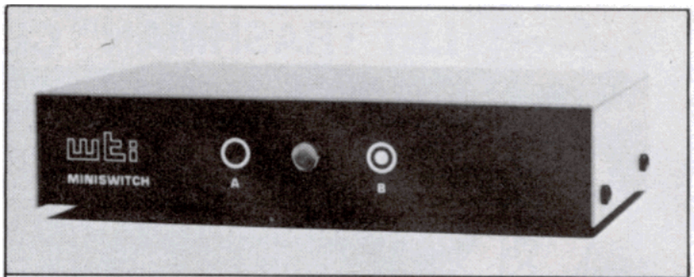
$24.95

ORDER FROM YOUR LOCAL SOFTWARE DEALER, OR CALL NOW, TOLL-FREE:
(800) 824-7888, Operator 422

FOR ORDERS OR INFORMATION CALL:
(213) 764-3131, or write to us.

PROSOFT®
Dept. G, Box 560, No. Hollywood, CA 91603 ~95

TERMS: We accept VISA, MasterCard, checks, or even cash. Please add $2.00 shipping handling within U.S.A. or Canada, and $5.00 overseas. C.O.D. charge is $2.00 in U.S. only. We ship within one day of receiving orders. In Calif., add 6½% sales tax.



The Mini-T-Switch lets your peripherals share a common component or a CPU I/O port.

ettes alone are $5.95. Samples, two Floppy Pockettes, are available for $1. They are sold by CDC Serious Software, 13715 Vanowen St., Van Nuys, CA 91405, 213-997-9692, 213-780-2958.

Reader Service ✓ 559

Hi! What's Your Name?

Now you can produce customized name tags, table tents, and labels for any occasion with NAMETAGR. The program's data entry routine is highly efficient. It lets you skip, back-up, or ditto. You can select your name tag format from the file or customize your own. The program takes full advantage of the dot-addressable feature of dot-matrix printers by letting you design and print your own logo.

The package includes two 5¼-inch disks, 30 bulldog holders, 400 pin-feed name badge stock, 240 labels (3½ by 1 inch), 120 labels (4 by 2 inches), and a manual. Available for Models I and III, it costs $79 from ETS Center, P.O. Box 651, Willoughby, OH 44094, 216-946-8479.

Reader Service ✓ 561

Mini Switch

The ABS Mini Switch ends the hassle of plugging and unplugging your data cables for different operations. It provides push-button switching between any two ports (A and B) and a common port. Typical applications include two terminals switching between one modem, one terminal switching between two computers, and a computer switching between two printers.

Available from Western Telematic Inc. (2435 South Anne St., Santa Ana, CA 92704, 714-979-0363), the ABS Mini Switch costs $89.

Reader Service ✓ 568

Minuteman: An emergency power supply and voltage regulator.

Shot Heard 'Round the World

Minuteman is a new uninterruptible power supply from Para Systems (2409 D Ave. J, Arlington, TX 76011, 817-640-0837). It is designed to protect your microcomputer from power failures, brown-outs, and voltage surges. When a power fluctuation does occur, Minuteman immediately switches from A/C power to its own emergency battery power for up to 15 minutes. This gives you time to save your data and shut down your system without losing valuable input and programming hours. The unit automatically switches back to A/C when power is restored.

Minuteman also features voltage regulation and surge protection as well. The unit provides standby power for microcomputers and word processors with a power rating of 200 watts or less. It sells for $395.

Reader Service ✓ 563

Bringing It All Together

The Micro Matrix II is a line-controlled central exchange for up to eight RS-232C, current loop, and TTL (transistor-transistor logic) devices. It features a patented circuitry that lets you transfer data anywhere in your network as long as the devices are under software control.

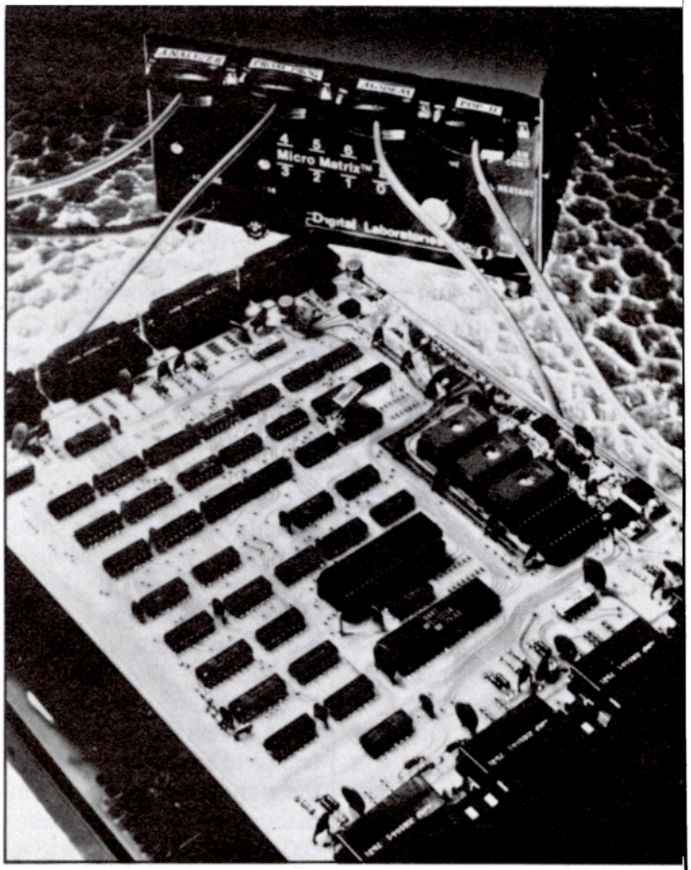
Providing 64 memory-mapped connection points, set-up is simplified by firmware that includes prompts, a command menu, connection displays, and ports handled by user name. The Micro Matrix II uses 1K RAM of non-volatile storage for 16 different switching arrangements. All 64 connection changes occur within 7 microseconds with no switching glitches or transmission interruptions.

Priced at $795 for the single-board configuration and $995 for the complete enclosure, the Micro Matrix II is sold by Digital Laboratories Inc., 600 Pleasant St., Watertown, MA 02172, 617-924-1680.

Reader Service ✓ 565

Customized Adventure Software

Give your friend or loved one a unique and personal gift on the next special occasion: bitCard. BitCards are challenging text and graphics adventures whose themes re-



The Micro Matrix II: A software-controlled data switch that connects RS-232C, Current Loop, and TTL devices.

late to a specific holiday or event. Their unique feature is that you can order them custom-programmed. An individual lucky enough to receive a bitCard as a gift will find several personal references to himself as he proceeds through the adventure (e.g., his name found on a note uncovered in a locked drawer).

BitCards also deliver a personal holiday greeting to your friend(s) in whatever words you wish. Your message appears as part of a dazzling climactic animated graphics scene—the reward for successfully completing the adventure (impatient players can jump to this final scene by using a password).

Available for the Models I, III, and Color Computer, bitCards cost $16.95 each from bitCards, 120 South University Drive, Suite F, Plantation, FL 33317, 305-473-4741. BitCards are available for Christmas, Valentine's Day, birthdays, graduations, and other holidays and milestone occasions.

Reader Service ✓ 556

Put It on Display

Bush Industries Inc. (312 Fair Oak St., Little Valley, NY 14755) makes a versatile desk for your personal computer. Finished in Arcadian hickory vinyl veneer, the desk (Model CT-100) measures 34 by 33 by 18 inches with a full width platform above the desk surface for your monitor, disk drive, and printer. A storage shelf below the work surface, also full width, is designed to hold manuals, software, and accessories.

It has casters for easy maneuverability, and you can use the desk as a student work center or typing desk as well. It is priced at $69.95. For further information, contact Felderman/Sharp Communications Inc. at 216-464-7252.

Reader Service ✓ 558

Model 100 Memory

The IM-100 is an 8K RAM memory module that plugs directly into the existing sockets of your Model 100 with no modifications to the system. You can add up to three of these plug-in modules to upgrade the portable computer to a total of 32K RAM memory.

Each memory module sells for $75. If you buy two or more units, the price is reduced to $70 each. For further information, contact Holmes Engineering, 5175 Green Pine Drive, Murray, UT 84107, 801-261-5652.

Reader Service ✓ 554

Raiders of the Lost Pyramid

An adventurer, you are looking for the hidden buried entrance to a pyramid that holds priceless treasures hidden for thousands of years. Your only tools are a crude map (a cube with ancient markings), a partial hieroglyphics dictionary, and a special navigation box. After you discover the entrance, you'll meander through rooms and chambers filled with one death trap after another. You are to unravel all the mysteries of this ancient pyramid and find all the treasures. The catch? Get out alive!

Infidel is the newest prose adventure game from Infocom. Its realistic environment is a result of the research the adventure's author did on pyramids and Egyptian history. The game features Interlogic, a development system that lets you use complete sentences instead of the standard two-word commands. Infidel recognizes a 600-word vocabulary, so you shouldn't have much trouble communicating.

Infidel costs $49.95, and is

See List of Advertisers on Page 291

available for both the Models I and III. For further information, contact Infocom Inc., 55 Wheeler St., Cambridge, MA 02138, 617-492-1031.

Reader Service ✓ 550

VisiCalc Mentor

Master the VisiCalc electronic spreadsheet in a matter of hours with *VisiCalc Programming: No Experience Necessary*. It is a self-instructional disk and guide that gives you hands-on learning experience with VisiCalc.

You begin with the basics and progress through advanced functions while simultaneously building a working Profit Analysis model. A self-pacing Checkpoint system lets you correct errors or review any part of the tutorial at any time. In addition to the Profit Analysis model, there are five other program templates included: Time Management, Future Cash Flow Analysis, Portfolio Valuation, Retirement Planning, and a Loan Repayment Scheduler.

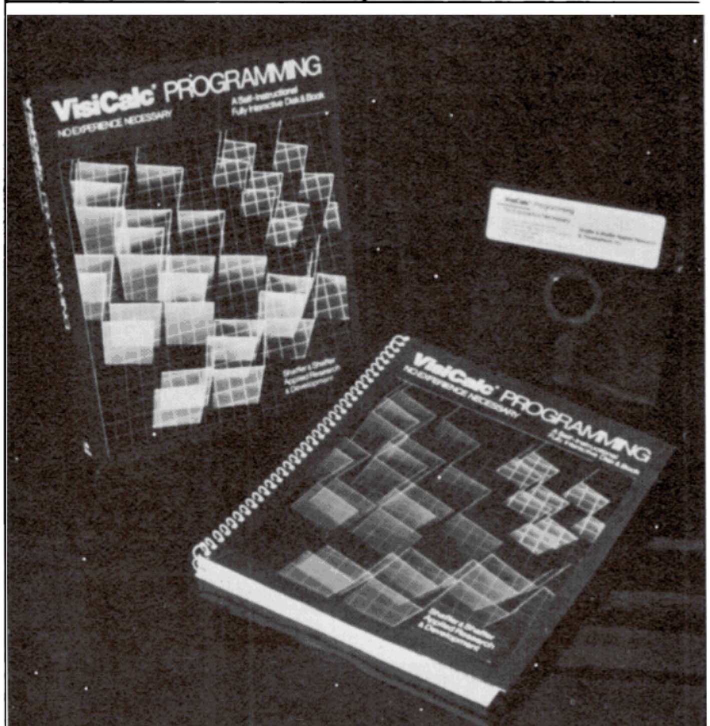
The slipcase package contains a spiral-bound 236-page manual with the sealed disk affixed to the inside front cover, and a command reference card. The Model III version has a suggested retail price of $59.95. It is sold by Little, Brown and Company, 200 West St., Waltham, MA 02154, 800-343-9204, 617-890-0250.

Reader Service ✓ 555

Teach Your Computer to Lisp

A fast machine-language Lisp system is now available for you Models I, III, and 4 owners out there. The system is designed to satisfy virtually all your needs, so it has features such as pixel graphics and floating-point routines in addition to a full range of Lisp functions.

The Lisp Interpreter features full recursion, function tracing and error trapping,



Learn VisiCalc in a few short hours with this self-paced guide.

WE'RE SERIOUS ABOUT FUN!

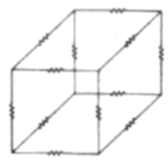
Software for Radio Shack's TRS-80 MODELS II/12/16

- GAMES • ADVENTURE
- BUSINESS • LANGUAGE
- EDUCATION

CALL OR WRITE FOR FREE BROCHURE AND MORE INFORMATION

Rizzo Data Systems Corp.
577 Burlington Rd., P.O. Box 458
Bridgeton, NJ 08302-0356
609/451-7979 ✓518

CIRCUIT SOLVER I



You don't have to be an engineer to make productive use of this program. A large clear manual leads you through the use of Circuit Solver using sample problems which have practical application.

Features

- Simple Circuit Entry
- Circuit Listing
- Circuit Storage & Retrival
- Easy Circuit Editing
- Supports Op-Amps
- Many Useful Sample Circuits

ORDER #		
CSI-100C	PET/CBM/64*	$34.95
CSI-101C	APPLE*	34.95
CSI-102C	TRS-80*	34.95

Include $2.50 shipping and handling.

*PET/CBM/64, APPLE and TRS-80 are * Trademarks of Commodore, Apple Computer and Tandy Resp.

✓432

SHS SOLUTIONS
1430 N. LATROBE
SUITE 2A
CHICAGO, IL 60651

- Raises & tilts TV monitor for easy viewing
- Allows CC keyboard to slip under monitor for more desk space
- Matching silver-gray fiberglass
- $37.50 + $2.50 shipping

"80" DEMI-CASES

- Models for LNW or MDX I & II Boards
- $32.50 + $2.50 shipping

N.Y.S. Residents Add 7% Tax ✓174

SYRACUSE R & D CENTER
Box 125, Dewitt, N.Y. 13214
"Specializing in Electronic Packaging"

DISCOUNT COMPUTERS

100% RS COMPONENTS, NO FOREIGN DRIVES OR MEMORY — FULL WARRANTY

24K MODEL 100	799.00	DMP 100 PRINTER	$ 299.00
16K MODEL 4	799.00	DMP 200 PRINTER	599.00
64K MODEL 4, 2DR, RS232	1579.00	DMP 500 PRINTER	1029.00
80K MODEL 12, 1DR	2499.00	DMP 2100 PRINTER	1599.00
80K MODEL 12, 2DR	3149.00	DWP 410 DAISY	1095.00
256K MODEL 16B, 1DR	3898.00	DWP DAISY WHEEL II	1499.00
256K MODEL 16B, 2DR	4510.00	12 MEG HD MODEL II/12/16	2369.00
256K MODEL 16B, 1DR W/HD	5439.00	ALL RS SOFTWARE	20% Off

CASHIERS CHECK OR MONEY ORDER MUST ACCOMPANY ALL ORDERS.

(817) 825-4027

NOCONA ELECTRONICS • Box 593 • Nocona, TX 76255 ✓232

two/sixteen magazine

The journal for business, professional, and scientific members of the TRS-80 community

THE ONLY MAGAZINE FOR MODEL II & 16 USERS

two/sixteen magazine
131A East Orange St.
Lancaster, Pa. 17603
✓522 **(717) 397-3364**

NODVILL DIET PROGRAM

$69.95

"TAKE A BYTE"©

MAINTAIN, LOSE OR GAIN WEIGHT
With your TRS-80 Double Density Disk Model I/III/4
and
The NODVILL DIET PROGRAM "Take a Byte"

you can quickly and accurately:

- **Calculate** caloric and nutritive food intake
- **Evaluate** nutritive value of your diet
- **Compare** daily diet to individual RDA Chart
- **Create** personalized daily meals and menus
- **Plan** varied daily menus based on sound nutrition
- **Save** records of daily meals and menus for future planning
- **Print** nutrition charts, food, meal, menu and grocery lists

"Take a Byte" is a Modular BASIC Program

MAIN Program Menu
(1) Recommended Daily Dietary Allowance (RDA) CHART
(2) 733 Expandable Random Access FOOD LIST Data File
(3) 28 Nutritional MEAL LIST Data File Examples
(4) 7 Balanced DAILY MENU LIST Data File Examples
(5) GROCERY LIST Program Module
(6) 25 Page USER'S MANUAL including Charts and Tables

ALL Data Files can be EXPANDED and MODIFIED systematically and flexibly to reflect your personal diet, your choice of diet book, or your doctor's suggested diet.

For more Information: Call: (203) 431-6449
To Order: Mail a check or money order for $69.95 to: ✓54

NODVILL Software
24 Nod Road
Ridgefield, Conn. 06877

TRS-80 is a Trademark of the Tandy Corporation

ALLOWS UNLIMITED COPIES OF MOST RADIO SHACK PROGRAMS INCLUDING SCRIPSIT AND VISICALC

Comes on a disk

No Programming Experience Necessary

ONLY $15.00

BACKUP

TOLL FREE 800-835-0071

In Kansas 316 665-3611

✓142

SALES DATA, INC.
526 E. 4th
HUTCHINSON, KANSAS 67501

property lists, 180 predefined Lisp functions, trigonometric and exponential functions, visual display capability, abbreviated quoting, and much more. The Lisp Editor is a standard, expression-oriented editor that features general pattern matching for searching and substituting.

The system runs on TRSDOS, LDOS, and NEWDOS, and includes a formatted disk with a Lisp Interpreter, a Lisp Editor, a differentiator and algebraic simplifier, a manual, and a poker player. Priced at $79.95, the system requires one disk drive and 48K. It is sold by Artificial Intelligence Technologies, 2121 NE 152nd St., Redmond, WA 98052, 206-644-3068.

Reader Service ✓ 572

Volksmodem: An inexpensive direct-connect modem.

Dial Direct!

Volksmodem is an inexpensive direct-connect modem from Anchor Automation Inc. (6913 Valjean St., Van Nuys, CA 91406, 213-997-6493). This means that you no longer have to worry about missing a phone call just because your phone lines are tied up to an acoustic modem. With Volksmodem, a simple flip of a switch lets you either go on line or make and receive phone calls.

The modem has a suggested retail price of $69.95. You have to purchase computer adapter cables separately. Both the RS-232 and Radio Shack adapter cables are priced at an additional $12.99. You can buy Volksmodem at major department stores and with mass merchants nationwide.

Reader Service ✓ 560

How to Sell Your Programs

You've slaved for months without sleep, living on coffee and root beer, trying to finish your program. Now you're done, and you're ready to stand the computer industry on its ear. But where do you go? Whom do you show your priceless program to? Programmers' Pipeline may be the answer.

Programmers' Pipeline is an electronic listing service

that posts a description of your software program to dozens of companies looking for new software to add to their product lines. They post your description within minutes after receiving it, day or night. They reinforce this online service by sending you a periodic hard-copy follow-up mailing of your listings.

Programmers' Pipeline deals directly with software companies. They have no agents, commissions, or hidden fees. The initial program listing service under one program category costs $20. Additional program categories cost $5 each. An optional blind-box service that receives and forwards responses to your program description is available for an additional $10.

If you'd like to see the Programmers' Pipeline in action, call via modem: 213-516-9432. The system asks you for your first name; respond with Pipeline. It then prompts you for your subscriber number; enter 80 Micro. It's that easy!

For further details, contact Programmers' Pipeline, P.O. Box 666, Glendora, CA 91740, 213-914-4317. As a bonus, if you mention *80 Micro*, you'll receive a free copy of *Checklist for Program Authors*. Merry Christmas!

Reader Service ✓ 573

Blushing Computers

Add vivid color to your Model I or III with CHROMAtrs, a color graphics peripheral from Micro Control Systems Inc. (1590 Broadway, Hewlett, NY 11557). CHROMAtrs uses 71 easy commands in addition to standard Basic commands to let you quickly plot points and lines, do 3-D rotations, translations, and create sprite graphics.

CHROMAtrs offers you 15 brilliant and distinct colors with which you create your own arcade-type games. It also lets you add joystick and paddle game action, as well as exciting sound effects.

The peripheral attaches easily to your Model I or III without affecting your Radio Shack warranty. Each unit costs $199 and comes complete with documentation. You can get more information from the Promotional Services Group at 516-781-0153.

Reader Service ✓ 578

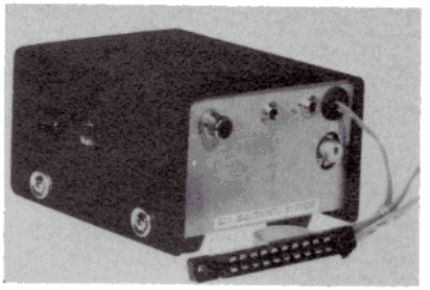
Sharing Your System

Ball Technical Services now makes full multi-user accounting software for your Model 16 that lets you operate up to three terminals simultaneously with the same accounting program. The available programs (Accounts Receivable, Accounts Payable, General Ledger, Payroll, and Order Entry/Inventory Control) all operate under the RM/COS multi-user operating system. This lets you print from remote terminals.

The programs have full record-locking capability. If a second operator requests a record that is being processed by someone else, it's flagged to say, "Wait, somebody's looking at the record."

RM/COS requires 256K of RAM to support three users and an additional 63K for each added terminal. It is three times faster on your Model 16 than on any other multi-user operating sys-

Now - Add External Monitors To Your Model III



Plugging in VIDEO MOD lets you plug in your TV or extra Monitor for Big Screen Viewing and Remote Display.

Great addition for displaying Home Games, Classroom Instructions, and presenting information at Business Meetings.

Standard features include:
- Monitor and/or TV hookups.
- Full 64 character wide screen display.
- Works with any program or operating system.
- No cutting or soldering required for installation.

EJB ELECTRONIC SYSTEMS
2902 Eggert Road
Tonawanda, N.Y. 14150
(716) 837-9411

Mfg. of Specialized Computer Products

VIDEO MOD only $169.95
ship. & handling $2.00
NY residents add 7% sales tax

114

PROFESSIONAL SOFTWARE **TRS-80® MODEL II & 16** NOW AVAILABLE!

DISK SORT 2.0 FOR **HARD DISK** OR FLOPPIES NEW! UP TO 15 + % FASTER **$119.95***
TRSDOS® 2.0a & 4.1 COMPATIBLE. FILL IN THE BLANKS & GO TO IT
WILL SORT ANY RANDOM FILE. OUTPUT OPTIONS: TAGS, TAGS & KEYS, OR COMPLETE FILE
1,000 REC's. SORTING 15 BYTES, INCLUDING ALL DISK I/O = 2 MIN. 33 SEC's (FLOPPY)
HARD DISK IS MUCH FASTER! NON-STOP JOB STREAM EXECUTION

BASIC CROSS REFERENCE • SAVE HOURS! **$59.99***
• SEE USED LINE #'s/NAMES AT A GLANCE • PRODUCES NICE PROGRAM LISTING
• OTHER OPTIONS INCLUDE MAKING NEW PROGRAMS, DROPPING REMARKS

CUSTOM PROGRAMMING/SYSTEMS DESIGN

DOC. ONLY $10.00 DEDUCTIBLE ON PURCHASE

*PLUS POSTAGE AND HANDLING. PRICES SUBJECT TO CHANGE WITHOUT NOTICE

TRS-80 & TRSDOS ARE REGISTERED TRADEMARKS OF TANDY CORP.

GOOD-LYDDON DATA SYSTEMS 186
5486 RIVERSIDE DR. • CHINO, CA 91710 • (714) 980-4563

M/C VISA ACCEPTED

DO YOU OWN A STATIC MONSTER?

Static electricity is probably the most persistent problem faced by the average personal/micro-computer user. Static can cause erratic or erroneous data transmission, loss of program, damage to discs and accessories, and even computer component failure. All this is costly, annoying, and unnecessary!!!

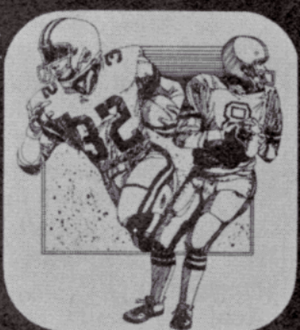
The **Static Control Computer Mat** puts static protection at your fingertips where it belongs. Simply touch the mat before touching the computer or any accessories and static is safely drained off to the mat and is sent hurrying off to ground before it can become a problem. There is no ZAP or spark and by periodically touching the mat while operating the computer, you will remain static safe. It is the effective economical approach to static control in the home or office.

The Static Control Computer Mat.

Only **$89.95** Check, M.O., C.O.D.

DELTA ELECTRONICS, INC.
Industrial Electronics Distributor
174 Main St., Norwalk, CT 06851
(203) 847-7285

DEALER INQUIRIES WELCOME! 346



PIK'EM 1983

67% ACCURATE

AGAINST POINTSPREAD
1982 SEASON
INCLUDING
AN OUTSTANDING
13 OF 15
PLAY-OFF GAMES

ORDERS CALL COLLECT
PHONE LINES OPEN 24 HOURS

VISA C.O.D. MasterCard

NAME ______
ADDRESS ______
CITY ______ ST ______ ZIP ______
PIK'EM 29^{95} ☐ STAT SER 35^{00} ☐
MI ☐ MIII ☐ APPLE ☐ IBMPC ☐
VISA ☐ MC ☐ PAY ENC ☐
CARD EXPIRES

A COMPLETE PRO FOOTBALL PREDICTION PROGRAM FOR THE 1983 NFL SEASON—INCLUDING PLAY-OFFS

★ ★ ★ ★ ★

20 MIN DATA INPUT PER WEEK WILL GIVE YOU THE EDGE

$29.95

INCLUDES: DISC DOCUMENTATION

TRS-80 MODEL I / III
IBM PC
APPLE
REQUIRES 32K

*STATISTICS NEEDED TO RUN THIS PROGRAM ARE AVAILABLE IN LOCAL NEWSPAPERS
OR:
MARATHON WILL FURNISH ALL NECESSARY STATS IN AN EASY TO READ ONE PAGE FORMAT FOR 19 WEEKS.
STATS ONLY $35.00

— FEATURES —
- DISPLAYS PREDICTED SCORES 219 GAMES
- DISPLAYS ACCUMULATED STATS/TEAM
- DISPLAYS AVERAGE STATS/TEAM
- HARDCOPY PRED — STATS — AVE
- AUTOMATIC OPPONENT INPUT
- AUTO STORE DATA — NO 'SAVE'
- UPDATEABLE FOR 1984

IF PROGRAM IS BOUGHT AFTER SEASON STARTS — ALL PREVIOUS STATS NECESSARY TO UPDATE WILL BE INCLUDED — NO CHARGE.

OTHER PROGRAMS
• PIK'EM 1984 FOR USFL •
• HOME BUDGET PLANNER •
• LETTER WRITER •

Marathon Software Dept M
P. O. Box 1349
Jacksonville, Texas 75766
Phone (214) 586-8212 241

See List of Advertisers on Page 291

tem—with half the memory. And it's 15 times faster than TRSDOS.

You can update existing installations of Radio Shack's three-disk accounting programs (26-4601, 26-4603, 26-4604, 26-4605, and 26-4607) to the multi-user RM/COS version for $300 per program. Or buy RM/COS separately for $750, with each program costing $599, or purchase RM/COS with all five accounting programs and Sales Analysis for $3,150.

For additional information, contact Ball Technical Services, 211 North First St., Mt. Vernon, WA 98273, 800-426-2070, 800-545-6244 (WA), 206-336-6605 (Canada).

Reader Service ✓ 576

Yo, Ho, Ho and a Bottle of Rum

Dancing Sailors is the dual-pen version of the DMP-40 plotter from Houston Instruments. It enables you to generate two-color plots or plots using different line widths without having to intervene. A simple command directs the plotter to place the alternate pen in plotting position. Since both pens are carriage mounted, pen changes are fast and easy.

The DMP-40-2 plotter costs $895 from Houston Instruments, 8500 Cameron Road, Austin, TX 78753, 512-835-0900, 800-531-5479. Both RS-232 and Centronics parallel versions are available.

Reader Service ✓ 581

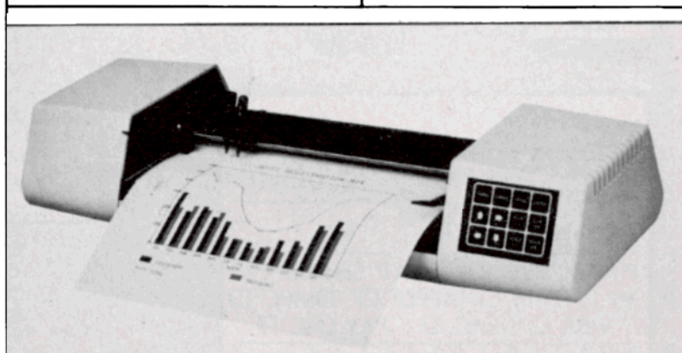
Let's Draw

Creative-Art is an advanced utility to design graphics on your Model III. The program has over 25 commands to let you draw circles, lines between points, rectangles, frames, and a single-key screen invert. Other commands include filling in and erasing parts of the screen, storing the screen in one of five pages of memory and displaying them later.

The program also handles text. An alphanumeric mode lets you place text anywhere on the screen. This is helpful when you generate reports and graphs. You can store screens on either cassette or disk. The package includes a sample program to retrieve the screen from disk or cassette for use with your own program.

Creative-Art is available for the Model III for 16K, 32K, and 48K cassette systems, and for 32K and 48K disk systems. Both the 32K and 48K versions store more pages and handle more points. All versions cost $30 for cassette and $40 for disk. Contact Creative Software Enterprises, Route 1, Box 222-A, Ana, IL 62906, 618-833-7797. All registered owners receive free updates for one year.

Reader Service ✓ 577



The Dancing Sailors Plotter gives you two-color plots.

No More Headaches

Solve your complex problems with Brainstormer, a powerful software program that helps generate potential solutions. The program works by building a description of a problem in terms of the themes and variations which affect its solution. Brainstormer analyzes the description of the problem to generate ideas for potential solutions. You refine the process by controlling the occurrence of particular themes and variations until the program produces a sufficient quantity of potential solution strategies. Brainstormer generates up to 10 billion idea probes for any user-specified problem.

Some potential applications for Brainstormer include increasing flexible thinking, discovering new products, targeting new markets, and exploring organizational problems. Brainstormer is available for Models I, III, and 4. It costs $50 for a single machine, $100 for 2–10 machines used by a single organization. For information, contact Soft Path Systems, c/o Cheshire House, 105 North Adams, Eugene, OR 97402, 503-342-3439.

Reader Service ✓ 567

Video Upgrade

Expand the video display on your Model III to 24 lines by 80 characters with Holmes VID-80 video upgrade. It is a plug-in printed circuit board that not only expands your display to 24 by 80 but that also lets you use the CP/M 2.2 operating system.

VID-80 features include 56K of user memory (64K RAM), the ability to read and write numerous disk formats, 8-inch drive support, CP/M compatibility, and UCSD P-system compatibility.

The VID-80 board retails for $279.50, and the CP/M 2.2 operating system sells for an additional $120. Other options, such as an additional 64K of memory (112K total) and the CP/M 3.0 operating system are also available. The board requires no trace cuts or soldering for installation. For further information, contact Holmes Engineering at 5175 Green Pine Drive, Murray, UT 84107, 801-261-5652.

Reader Service ✓ 552

New Vision

Get rid of your screen's glare and see what you're programming or writing with Panelgraphic's Video Filters. Treated with an anti-glare coating, the filters make a dramatic improvement in the readability of your computer screen. Characters boldly stand out on a professional-looking dark background without reflection.

The filters are easy to install with self-adhering tabs. They cost $19.85 each from Panelgraphic Corp., 10 Henderson Drive, West Caldwell, NJ 07006, 800-222-0617, 201-227-1500. Major credit cards are accepted for large orders.

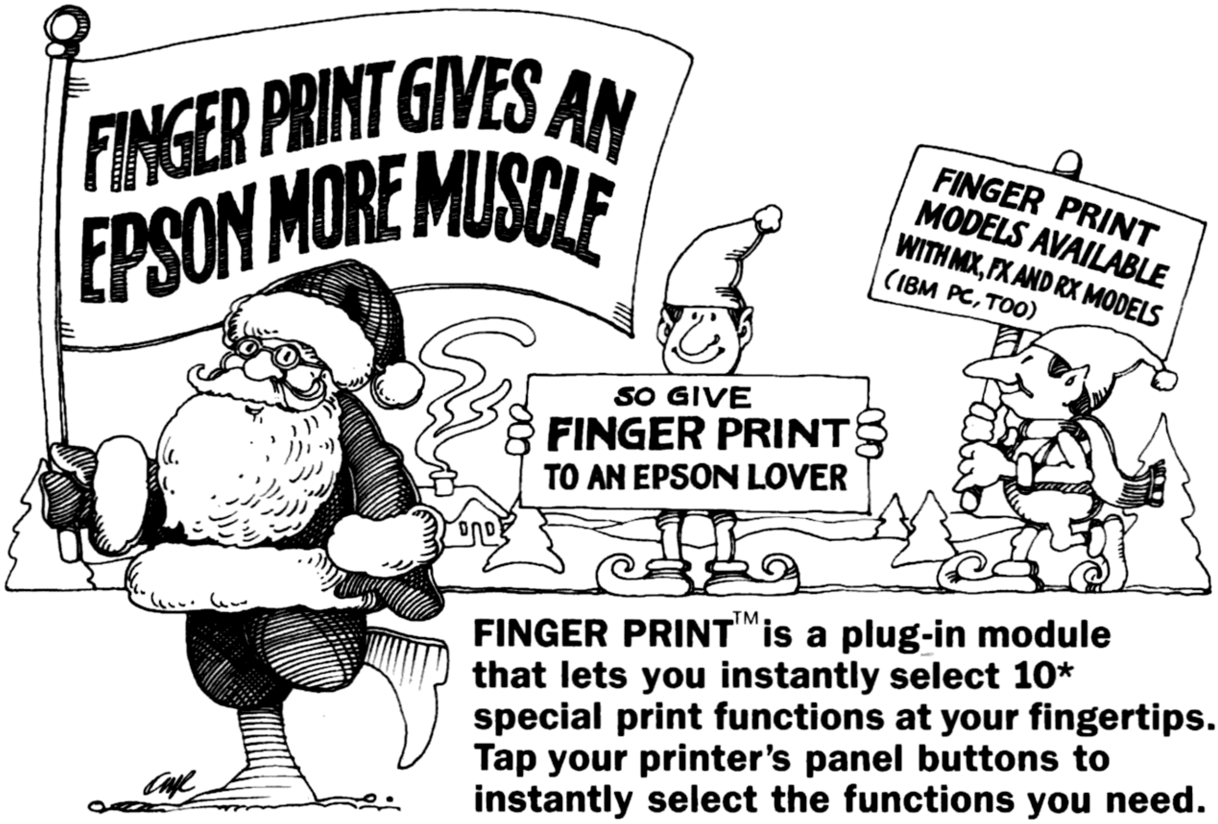
Reader Service ✓ 570

Terminal Emulator Program

All you Models II, 12, and 16 owners can now communicate with Digital Equipment Corporation (DEC) host mainframes over asynchronous (dial-up) lines with the VT52 Terminal Emulator program. The emulator supports all VT52 features and permits the use of any program requiring full-screen cursor control.

Common applications include interactive full-screen text editing, electronic mail, and the more sophisticated games. The emulator functions as a normal dumb terminal when communicating with non-DEC hosts.

The VT52 Emulator costs $175, including a User's Manual and a keypad editing



FINGER PRINT™ is a plug-in module that lets you instantly select 10* special print functions at your fingertips. Tap your printer's panel buttons to instantly select the functions you need. But, seeing is believing...

Want to use compressed to get more columns on your spreadsheet? You got it!

Need a double-wide heading? Easy!

Want to emphasize print on your final draft? No sweat!

Tired of printing on the perforation?

With automatic perforation skipover, it doesn't happen!

Want to use italics to make a paragraph stand out? Simple!

Need Fine Print for a contract or footnotes on a report? With Fingerprint just tap the buttons!

Even combinations like double emphasized italics are no problem!

That's just a sample of how FINGER PRINT turns your Epson MX 80/100 or IBM PC printer into a hard working printer...and FINGER PRINT has Buffer Clear and Left Margin Control, too!

FINGER PRINT will not interfere with normal printer operation...and installs easily, without soldering. It comes with complete operating/installation instructions, control panel reference label, and it's WARRANTED FOR 1 FULL YEAR.

*For Graftrax-Plus equipped printers. Italics and Fine Print cannot be selected on non-Graftrax-Plus models.

$59.95 Look for FINGER PRINT at your local computer dealer. For the name of a dealer near you, or additional information, call (213) 914-5831. To order directly, call toll free: 800-835-2246, Ext. 441. MC/Visa/COD orders accepted (include $1.50 s/h...CA residents add sales tax).

FINGER PRINT T.M.

DRESSELHAUS COMPUTER PRODUCTS

We make technology easy to live with.

837 E. Alosta Ave., Glendora, CA 91740

We welcome dealer inquiries

353

See List of Advertisers on Page 291

template. It is available from IMPACC Associates, P.O. Box 93, Gwynedd Valley, PA 19437, 215-699-7235.
Reader Service ✓ 582

Get Rid of the Surge

The SL Protector from Transtector Systems (East 5250 Seltice Way, P.O. Box 1299, Post Falls, ID 83854, 800-635-2537, 208-773-1521) is a transient overvoltage protection device designed specifically for personal computers.

Measuring approximately 3 cubic inches, the SL Protector plugs directly into any standard 110-volt outlet and accepts all standard, three-prong plugs. It automatically cleans incoming electrical power lines of overvoltage transients and spikes, protecting your computer's solid-state components from damage. The unit responds to a transient in five billionths of a second, and features an instant reset to provide continuous protection.

The SL plug is UL listed, sells for $99, and comes with a 90-day replacement warranty.
Reader Service ✓ 553

Computer Carrying Case

There's no need to be afraid to take your computer with you on your next business trip as long as it is safely packed away in one of the computer carrying cases from Cases Inc. They offer two hardware cases, the Flight Form and the C-Series case.

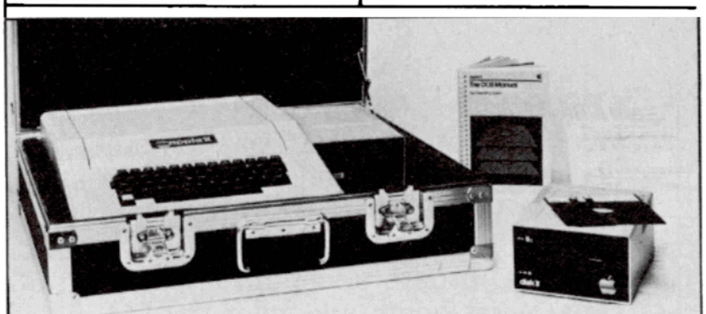
The Flight Form case is made of high-impact ABS plastic laminated to plywood. The outer panels of the case are slotted to lock into extruded aluminum tracking, forming a protective cage around your computer. The corners are finished with riveted steel for maximum strength. This case is designed to withstand the most rigorous handling. It could probably survive the famous baggage gorilla.

The C-Series case is designed for easy handling and convenience. It's made of Philippine mahogany covered with a tough vinyl outer lining. It provides your computer with protection in less demanding traveling or shipping environments.

Both cases feature form-fitted foam lined interiors and track mounts to hold equipment securely in place. They also accommodate books, papers, and software for your computer and you can specify custom interior configurations.

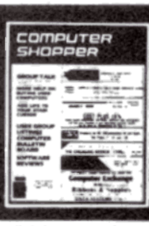
The C-Series is available in either black or brown. Prices start at $120. The Flight Line cases come in white, black, and royal blue. It costs $284 for the Model III, $318 for the Model II, and $331 for the Model 12. Accessories include special clasps, strapping, and even custom print-

Continued on p. 316



The Flight Form case: Now there's no excuse not to take your computer with you on your next trip.

SLICK PAGES? NO! GREAT DEALS? YES!



Computer Shopper pages aren't slick because they were designed for one purpose; to put buyers in touch with sellers at the lowest possible cost. This resulted in bargains on new and used equipment and software.

Individuals nationwide are able to list their pre-owned items for only a few dollars. This created hundreds of classified ads in over 100 big 11x14 pages.

And to make sure there were enough buyers for advertisers, the subscription price was set low, too!

So strike your first deal by taking advantage of the low subscription price, 12 issues for $15. That's a $8.40 discount off the newsstand price. Money back guarantee. **For faster service (MasterCard and VISA) Call 305-269-3211.** ✓39

COMPUTER SHOPPER
407 S. Washington Ave. • P.O. Box F587
Titusville, FL 32781

VITAMIN and MINERAL DEFICIENCY CHART

Your computer will analyze your answers to 192 questions concerning your body and then print a chart showing probable deficiencies of 33 vitamins and minerals plus protein and fat. (Such questions as, Do your eyes hurt when you go out into the sun? Is your hair beginning to grey? Do you sleep well? etc etc)

Comes complete with 50 questionnaires, 50 blank Nutritional program forms, instructions on completing the Nutritional forms from your computer generated chart, and an excellent instruction manual and a 5 1/4 inch disk. This program can put you into a business of your own selling nutritional programs.

ALLIED SYSTEMS CO.
P. O. BOX 245 TRONA, CALIF. 93562

FOR MODELS I/III/4 48K — 619 372-5355
C. O. D. AND MONEY ORDERS — $89.50
ARE RUSH ORDERS ✓140

TRS-80 III / IV

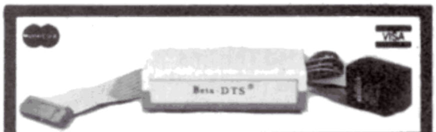
★ BOOKKEEPER ★
Perfect For Small Business
TIME DOME Has Preloaded File

★ SCRIPSIT-MAILER ★
LETTERS · ENVELOPES · LABELS
made by Scripsit

★ SERVICE-BILLER ★
Bill By Rate Chart
From-To Date

★ BOOKKEEPING MACHINE ★
Printed LEDGERS · STATEMENTS
1400 Accounts Receivable

$34.50 EACH PROGRAM
(714) 774-9383
H. A. K. WORKSHOP
9791 ORANGE AVE.
ANAHEIM, CA 92804 ✓287



DISK BACKUP
ONE MEGABYTE ON A C-60
MODELS I or III

Includes Disk to Disk Copy.
Auto Mixed Density Recognition.
Supports most Double Density Boards.
Configuration Mode For Your System.
Hassle Free Cassette Recorder Operation.
Set Volume Control to Maximum and Forget It.
Tape Disks at Computer Clock Speeds to 4 MHZ.
Auto Retrieval of Taped Disk with Formatting.
Will Backup Most Popular Dos Made Disks.
Includes Cassette Recorder Controller,
Power Supply, Computer Ribbon Cable, Disk & Manual.

Minimum Requirements: Model I, or III Computer, One Drive, Recorder with Cable, 32K of Memory.

NEW PRICES
Bata-DTS Mdl I — $109.95 & $ 3.50 H/S
Bata-DTS Mdl III — $113.95 & $3.50 H/S

Send for free copy of manual, "Reduced Size"

BETA ENTERPRISES INC.
14049 Settlement Acres Drive
Cleveland, Ohio 44142
(216) 362-6191 ✓25

INVESTMENT SOFTWARE

PORTFOLIO MANAGEMENT SYSTEM

Ideal for the private investor. Keeps track of up to 200 stocks, 100 options, 100 mutual funds, and 100 bonds. Produces a comprehensive portfolio breakdown and printed portfolio summary.......... $29.95

Investment Club Management System

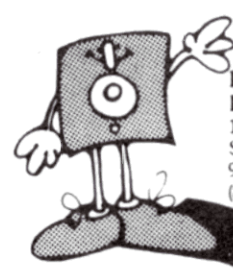
The perfect system for private investment clubs. Handles all current club holdings, including: up to 50 members, 10 accounts, 200 stocks, 100 options, 100 mutual funds, and 100 bonds. Produces complete reports including a portfolio summary, single member status, member listing, and overall club profit summary.............................. $69.95

Both programs are 100% machine language and require a 48K MOD I/III/IV, 2 disk drives, and any 80 column printer. Specify computer and printer type when ordering.

RENRAD SYSTEMS ✓246
3205 MESA DR. OCEANSIDE, CA 92054
(619) 721-6783
Cash, check or money order. Add $2.50 for postage and handling. COD add $3.00 extra. CA residents add 6% tax.

Dysan CORPORATION

Solve your disc problems, buy 100% surface tested Dysan diskettes. All orders shipped from stock, within 24 hours. Call toll FREE (800) 235-4137 for prices and information. Visa and Master Card accepted.



PACIFIC EXCHANGES
100 Foothill Blvd.
San Luis Obispo, CA
93401. (In Cal. call
(805) 543-1037.) ✓207

R.I.S.T. Inc. ANNOUNCES

SPEECH SYNTHESIZER SALE

ORDER NOW — ORDER NOW

$59.95 plus $4.00 sh/hd

ACT NOW Receive — Phrase Finder Program, Speech Synthesis Manual — **FREE**

TRS80 Models I, III
Apple II, II+
ZX80/81 TS1000
Com 64, VIC 20

Regularly $89.95
Now Only $59.95

Complete or Starter Kits Available

* * * 10 Day Money-Back Guarantee * * *
Send check or M.O. COD add $1.50 to: R.I.S.T. Inc. Dept. GZ10 3, P.O. Box 499 Ft. Hamilton Station Brooklyn, NY 11209 212-259-4934 ✓388
N.Y.S. RESIDENTS ADD 8¼% TAX

ATTENTION DEALERS!
RS-232 BOARD FOR TRS-80 MODEL 3 or 4

Functionally equivalent to the Radio Shack RS-232 Board. Meets EIA RS-232C specifications. Programmable baud rate from 50 to 19,200 baud. Programmable parity. Word length and stop bits. Full or half duplex operation.

IMAGE TECHNOLOGY INC. (303) 794-2074
5994 South St. Paul Way
Littleton, CO 80121

Volume discounts available.

One year warranty. ✓89

TRS-80
COMPUTER DISCOUNTS

- Factory Direct
- Best Prices Anywhere
- No Out-of-State Taxes
- 100% Radio Shack Warranty
- Free Price List

SCOTT TASSO ASSOCIATES
175 E North Delsea Drive
Vineland, N.J. 08360
800-257-0426
NJ 609-691-7100 ✓510

MODEL 4 DRIVE KITS $299

MODEL 4 STUFF
MODEL 4 COMPUTERS

all include an RS232

No charge for shipping on any Compukit Computers

64K one single headed drive .... *$1495.*
64K two single headed drives .... *$1695.*
64K two double headed drives .... *$1895.*
128K systems add only .... *$99.*

MODEL 4 64K UPGRADE $62.95
64K TO 128K UPGRADE $82.95

prime grade pre-tested RAM with instructions

MODEL 4/III RS232 KIT $69.95

completed and tested ready for installation

Model I, III, 4 green phosphor antiglare CRT Kit $89.

a complete new CRT, not a filter

MODEL 4 DISK DRIVE UPGRADE KITS

All of the Compukit Model 4 Disk Drive Upgrade Kits contain these features that other companies usually do not provide. Switching power supplies * Tandon disk drives * 64K of Model 4 RAM * Sound * and an Easy to Use Installation Manual. Requires only a screwdriver (no soldering) The no drive upgrade kit .... $299.
One drive upgrade kit .... $499.
Two drive upgrade kit .... $699.
Two double headed drive kit .... $899.

MODEL III to MODEL 4 UPGRADE $750.

Converts your Model III into a Model 4 (except for Cabinet and disk drives). Includes new keyboard, 64K RAM, Sound, and free installation (required). Ship us any working Model III, even if it's not all factory equipment, and get back a Model 4, in your case.

MODEL 4 SOUND UPGRADE $29.95

No soldering required, Includes instructions

COMPUKIT DOCTOR $29.95
Disk based diagnostic software package for the Model I, III, and 4

MODEL III DISK DRIVE UPGRADE KITS

All of the Compukit Model 3 Disk Drive Upgrade Kits contain these features that other companies usually do not provide. Switching power supplies * Tandon disk drives * 32K of Model III RAM * Compukit Doctor * and an Easy to Use Installation Manual. Requires only a screwdriver (no soldering).

The no drive upgrade .... $279.
One drive upgrade kit .... $479.
Two drive upgrade kit .... $679.
Two double headed drive kit .... $879.

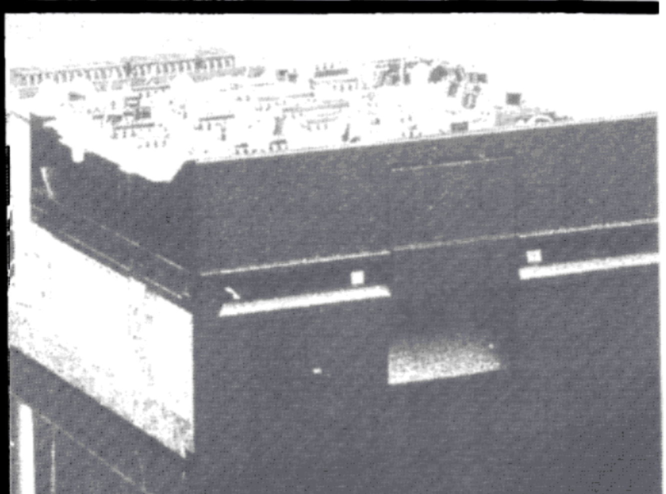
TANDON DISK DRIVES

Perfect for replacement or add on drive for Any 5″ drive system.

TM100-1 40 Track Single Sided .... $199.
TM100-2 40 Track Double Sided .... $299.
TM50-1 20ms trk-trk Single Sided Thinline .... $149.
TM50-1 6ms trk-trk Single Sided Thinline .... $169.
TM50-2 6ms trk-trk Double Sided Thinline .... $219.
TM55-2 microprocessor Double Sided Thinline .... $249.
Single case with extenders with drive above $49.95
Dual case with extenders for drives above $79.95
Two Drive Cable $24.95

64K COCO KIT $62.95

COCO DRIVE O
$379.
with a Tandon Thinline
SPECIAL INTRODUCTORY PRICE
DUAL DRIVES $529.



COMPUKIT CORPORATION

ordering information
We accept Visa, Mastercard, Wire Transfers, and Certified Checks for quickest shipping. Orders received on personal checks are held for clearance.

1-713-480-6000 order line 1-800-231-6671
16206D Hickory Knoll, Houston, Texas 77565

COMPLETE MODEL III/IV

HARD DRIVE $1095.

COMPLETE PRIMARY DRIVES
MODEL III & IV

5 MEG $1095.
10 MEG $1395.
15 MEG $1695.

MODEL I & LNW

5 MEG $1145.
10 MEG $1445.
15 MEG $1745.

COMPLETE SYSTEM FEATURES
TANDON DISK DRIVES
WESTERN DIGITAL 1002 DRIVE CONTROLLER
ONE YEAR WARRANTY
SWITCH SELECTABLE HOST
POWER ONE HD POWER SUPPLY
HD COOLING FAN
SIZE 8-1/4"X6-1/2"X13"
ALL DRIVES RATED AFTER FORMAT
COMPLETE READY TO PLUG IN (JUST ADD YOUR DOS)
SUPPORTS NEWDOS 80 V2.5, DOSPLUS 3.4f, 4.0, 3.5, IV, LDO and soon TRSDOS 6.0

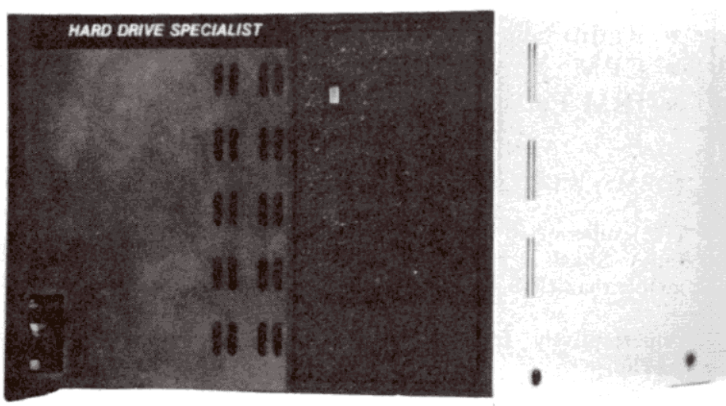
SECONDARY DRIVES

5 MEG $695.
10 MEG $995.
15 MEG $1295.

(READY TO PLUG INTO YOUR PRIMARY DRIVE)

NEW!

33 MEG MODEL PRIMARY HARD DRIVE (BY QUANTUM) $2395.
MODEL I & LNW $2445.



NEW PRODUCTS

READY FALL 83	*CALL ON AVAILABILITY*
21 MEG TAPE	*$649.*

Will Work on any Hard Drive System That uses DOSPLUS or NEWDOS

MODEL 3/4 BOOT ROM *$39.95*
ALLOWS YOU TO BOOT DIRECTLY FROM YOUR HARD DRIVE FOR DOSPLUS and NEWDOS SYSTEMS ONLY

MULTIPLEXER $995.

ALLOWS UP TO 4 COMPUTERS TO ACCESS A HDS HARD DRIVE
Includes Master Control Unit and Cable / Host Adapters for externally connecting 2 Computers

HARD DRIVE SPECIALIST a division of Compukit Corporation

1-800-231-6671 *1-713-480-6000* *16206D Hickory Knoll Houston, Texas 77059* 455

See List of Advertisers on Page 291

Continued from p. 312

ed logos. To place your order, or to inquire about a customized case, contact Cases Inc., 13102 Stone Ave. North, Seattle, WA 98133, 206-365-5210.

Reader Service ✓ 566

Computer Talk

Modems have been placed on a diet at Universal Data Systems! Their new LP modem series is entirely powered by your telephone line, eliminating the need for bulky, heat-producing power supplies or plug-in power modules. This results in a functional and attractive low-profile design that provides full modem features in a sleek, under-telephone package.

The five modems in this line each measure about 1 by 6 by 9½ inches, and use a standard RS-232 connector. They are designed for easy operation: A mode switch invokes answer or originate operation, and a talk/data switch simplifies transfer of a telephone call to the computer, allowing you to switch from voice to data communications.

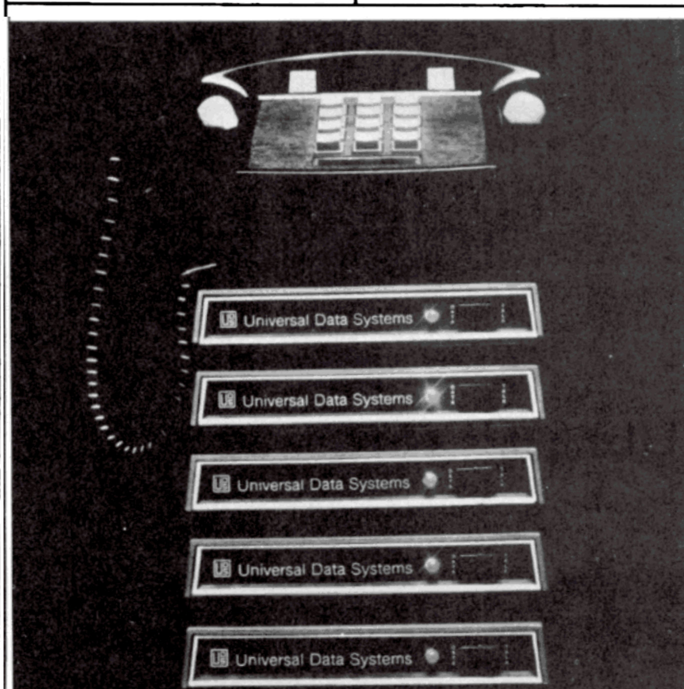
Prices for the LP modem series range from $145 for the 0–300 baud modem to $445 for the 1200 bits per second model. All are available from Universal Data Systems Inc., 5000 Bradford Drive, Huntsville, AL 35805, 205-837-8100.

Reader Service ✓ 578

More Memory

RAMPAK is a 64K RAM upgrade for the Model 4. It is available in two different versions: RAMPAK-I to upgrade a 16K to a 64K, and the RAMPAK-II to upgrade from 64K to 128K. The latter allows programs like MEM-DISK to take advantage of the Model 4's bank switching mode throughout the entire 128K. They cost $96 and $110 respectively.

For those of you who want to upgrade from 64K to 128K using your own 64K RAM chips, RAMPAK-PAL, a programmed array logic chip, is available separately. It costs $34.95.



The LP modem series from Universal Data Systems.

316 • 80 Micro, December 1983

MIKROKOLOR

TRS-80* MODEL 100 COLOR GRAPHICS

High resolution color graphics for graphs, charts, games, animation, business applications, teaching, scientific display. Allows full sized color text and graphics as featured in 80 Micro magazine, May & June 1983.

Display on any sized Color Monitor or Color TV with modulator.

256 x 192 Color Graphics - 15 colors plus transparent
3 dimensional Sprite planes - Simultaneous display of planes.
4 Modes of operation available:
1. - TEXT: 24 lines - 40 characters per line, 6 x 8 matrix, 256 user defineable characters.
2. - MULTICOLOR: 64 x 48 Color Graphics.
3. - GRAPHICS 1: 256 x 192 Color Graphics, 24 lines - 32 characters, 8 x 8 matrix, 2 colors per character.
4. - GRAPHICS 2: Same as Graphics 1 except 16 colors per character.

SPRITES: Active in all but text mode, 32 prioritized 3-D planes, 15 colors plus transparent, easily provides animated graphics with simultaneous display of all sprites. Composite Video Output (NTSC). Comes with sample programs and instructions. No hardware modifications necessary, plugs into expansion socket. Uses Texas Instruments TMS9918A Video Display Processor. On board RAM uses no system memory space.

User manual only - $5.00. Money Order, COD, checks or credit card. Personal checks must clear. COD add $2.00. Visa, Mastercard add 4%. Calif. residents add 6% sales tax. DEALER INQUIRIES INVITED.

$235.00 Assembled and tested.

$195.00 Kit, with instructions

*TRS-80 Trademark of Radio Shack/Tandy Corp.

ANDREASEN'S ELECTRONICS RESEARCH & DEVELOPMENT, Inc.

Technical Assistance: Box 5686, Vandenberg, Ca. 93437
To Order: 1548 Monterey St, San Luis Obispo, Ca. 93401 ph (805) 541-6398 ✓172

VSS-80

Viewgraph Slide System for Business, School & Home
For your TRS-80 Model I/III . . . $79.95

- ***Efficient Viewgraph Data Base***
- ***Hi-speed Graphics including 4 shadings***
- ***Supports many popular printers***

FUTURE PROJECTS CORP. (203) 775-3062
P.O. Box 11, Hawleyville, CT 06440 ✓528

COTTAGE SOFTWARE

PACKER: Machine language program that edits all or part of your Basic program to run faster, save memory, or ease editing. The 5 options include UNPACK—unpacks multiple statement lines into single statements maintaining logic, inserts spaces and renumbers lines. SHORT—deletes unnecessary words, spaces, and REM statements. PACK—packs lines into maximum multiple statement lines, including all branches. MOVE—moves line or blocks of lines to any new location on program. On 2 cassettes for 16K, 32K, & 48K
For TRS-80 Mod I or III Level II or Disk Basic . . . $29.95

SYSTEM TAPE DUPLICATOR: Copy your SYSTEM format tapes. Includes verify routines. The Model III version allows use of both 500 and 1500 baud cassettes speeds.
For TRS-80 Model I or III Level II $15.95

CASSETTE LABEL MAKER: A mini word processor to print cassette labels on a line printer. Includes 50 peel - and - stick labels on tractor feed paper.
For TR-80 Model I of III Level II & Printer $17.95

PRINT TO LPRINT TO PRINT: Edits your Basic program in seconds to change all Prints to LPrints (except Print@ or Print#) or LPrints to Prints. Save edited version.
For TR-80 Model I or III Level II $12.95

FAST SORT ROUTINES: for use with Radio Shack's Accounts Receivable, Inventory Control I, and Disk Mailing List Systems for Model I Level II Sorts in SECONDS! You'll be amazed at the time they can save. Supplied on data diskette with complete instructions.
FAST SORT for Accounts Receivable $19.95
FAST SORT for Inventory Control I $19.95
FAST SORT for Disk Mailing List (specify data diskette of cassette for 1 drive system) $14.95
ALL THREE ROUTINES $44.95

Prices subject to change without notice. Call or write for a complete catalog. Dealer inquiries invited. VISA and Mastercharge accepted. Foreign order in US currency only. Kansas residents add 3% sales tax.

On-line catalog in Whichita FORUM-80: 316-682-2113
Or call our 24 hour phone (316) 683-4811 or write:

COTTAGE SOFTWARE
614 N. HARDING
WICHITA, KANSAS 67208 ✓187

TRS-80™ is a trademark of Tandy Corporation

COMPUSETTE
AT WHOLESALE PRICES

ORDER TOLL FREE

Only $1.69*

SAME DAY PROCESSING

Only 36¢*

-Thousands of Satisfied Customers-

Computer Hobbyists·Public School Districts·Government Agencies

Blank Diskettes
Soft Sector·Single Sided·Double Density

10	20	50	100
$2.29*	$1.99*	$1.79*	$1.69*

*with custom Tyvek Envelope, Add 15¢

10-pack Custom Case $3.50 50-pack Case $22.95

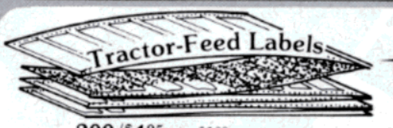
UPS SHIPPING: (10-20) $3.00 (50-100) $6.00

Blank Cassettes
100% Error-Free & Fully Guaranteed

Length	12	24	96	250*	500*
C-05	.69	.59	.49	.44	.36
C-10	.79	.69	.59	.49	.39
C-15	.89	.79	.69	.59	.49
C-20	.99	.89	.79	.69	.59
C-30	1.09	.99	.89	.79	.69
Boxes	.26	.21	.20	.16	.13

*250/500 Quantities without labels, With labels Add 4¢

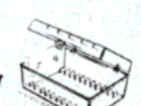
UPS SHIPPING (12-24) $3.00 (36-96) $6.00 (250) $12.00 (500) $18.00



200/$4.95 + $1.00 shipping & handling
1000/$18.95 + $3.00 shipping & handling
These labels are for cassettes

TAPE-PACK SPECIAL
6 Each: C-5, C-10, C-20, C-30
Only $16.95
+ $3.00 shipping & handling

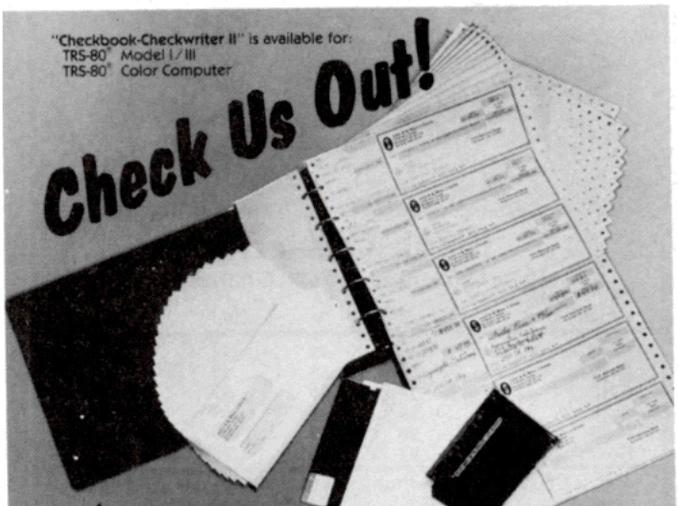
NEW **Tape Caddy**



Only $2.75
$1.00 shipping & handling

Micro-80™ INC.
2665 E. Busby Road
Oak Harbor, Wash., 98277 ✓132

Order Toll-Free
Orders ONLY 1-800-553-0035 Ext 80
1-800-528-6050 Ext 3005
Inquiries 1-206-675-6143



All upgrades are available from VR Data, 777 Henderson Blvd., Folcraft, PA 19032, 215-461-5300.

Reader Service ✓ 575

Options on Options-80

Options-80A is an advanced version of the Options-80 program, a tool to maximize investment return on stock options. This newer version includes Black-Scholes modeling, plus expanded handling of covered option writing and put option spreads. You can enter prices in fractional form, and print graphs in alphanumeric symbols so that you can use any printer. Plot or tabulate returns as a function of either actual or annualized percentage change in underlying share price.

Another feature of Options-80A is analyzing investment return on puts, calls, and spreads while taking account of commissions, risk, dividends, and time-cost money.

Both Options-80 and Options-80A are available for Models I, III, and 4 with disk drives. They cost $125 and $170 respectively plus $5 for post and packaging. Send your orders to Options-80, 15 Stow St., Concord, MA 01742.

Reader Service ✓ 569

Complete Protection

Now you can protect all your voltage-sensitive equipment from power surges and spikes with PTI Industries' Protector. The plug-in unit offers 0.1 nanosecond response time to surges and provides complete noise filtering from 100kHz to 100MHz.

The Protector is the first unit to offer brown-out notification when the line voltage drops to 100 volts. A reset button safeguards your equipment from initial line surges that inevitably follow a blackout condition.

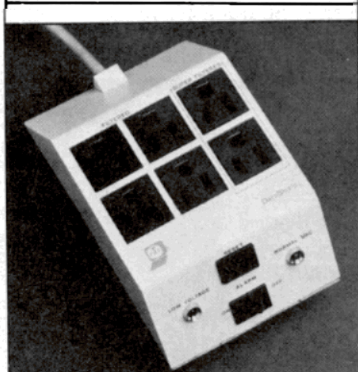
This surge, noise, and brown-out protector costs $69 from PTI Industries, 4740 Scotts Valley Drive, Scotts Valley, CA 95066, 408-438-3870.

Reader Service ✓ 564

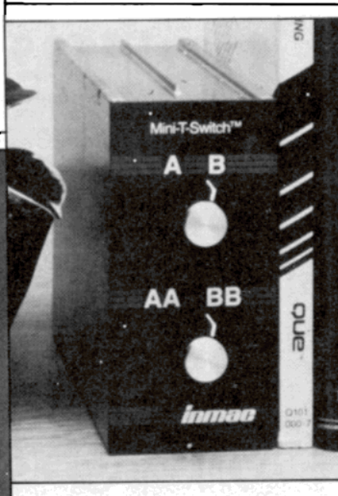
The Big Switch-a-roo

The Mini-T-Switch lets you reduce the number of modems and printers in your use by almost half, while still allowing you virtually endless system configurations and fast CPU–peripheral hookups. How? The unit has two T-switches, each of which can enable two peripherals to share a common third component or a CPU I/O port.

It functions either as a



A fast (0.1 nanosecond response time) surge and spike protector.



Switch between your computer and its peripherals with the AB Mini Switch.

CHURCH DONATIONS

307

Computer Program for TRS 80
Models I, II, III, IV, 12 & 16
IBM/PC

RECORDS, STORES & REPORTS

Gives printout of collections, totals to date as well as individual quarterly reports to congregation members.

•NOW AVAILABLE•

Additional modules to provide:
Church Directory
Word Processing
Attendance Record
Membership Profile
General Ledger

Radio Shack ★★★★★
Five-Star Rating

Designed to fulfill administrative & financial reporting and record keeping needs of churches.

CUSTOM DATA
Dept. AC
505-434-1096

301 Eleventh Street
P.O. Box 1869
Alamogordo, NM 88310

SEND AND RECEIVE
MORSE CODE

•TRIPLE SPLIT SCREEN
•SPEED: 1-200 WPM

This outstanding 4K machine language program for your Model I, III or 4 has a type ahead buffer active while copying. The computer generates the tone so only an audio amplifier is needed. A transmitter may be keyed. Send from keyboard, 8 messages, at random, or repeat. Received Morse is decoded, displayed, and printed. I/O is through the cassette interface. Morse cursors & other features.

satisfaction guaranteed or your money back

Cassette with instruction booklet only $19.95
postpaid. 16k required

ROGO Computer Products
4752 DeBeers Drive
El Paso, Texas 79924

343

SAVE BUCKS
ON VERBATIMS

$25.95 Box of 10
add $1.50 for shipping

5 YEAR WARRANTY

525-01

Single Sided / DOUBLE DENSITY SOFT SECTOR
VISA and MASTERCARD ACCEPTED

COMPUKIT Corporation

455

orderline 1-800-231-6671
1-713-480-6000

16206D Hickory Knoll Houston, Texas 77058

Subscription Problem?

80 Micro does not keep subscription records on the premises, therefore calling us only adds time and doesn't solve the problem.

Please send a description of the problem and your most recent address label to:

80micro®
Subscription Dept.
PO Box 981
Farmingdale, NY 11737

Thank you and enjoy your subscription.

DISKETTES
$15.95/BOX OF 10•

•5.25" SINGLE SIDED, SINGLE DENSITY
MANUFACTURED BY ONE OF THE
WORLD'S LARGEST PRODUCERS
ALL DISKETTES ARE SOFT SECTORS
EACH DISKETTE IS CERTIFIED ERROR FREE

5.25" SSDD $18.95/BOX OF 10
5.25" DSDD $22.95/BOX OF 10

TWO BOX MINIMUM ORDER
PLEASE ADD $1.50/BOX SHIPPING & HANDLING
DEALER AND CLUB INQUIRES INVITED

(312) 490-9239
SHEPHERD MARKETING

P.O. BOX 941339
VISA SCHAUMBURG, IL 60194

IL. RES PLEASE ADD 6% SALES TAX

34

GRADE BOOK

- Disk program (one disk required)
- Simple comprehensive
- Teacher designed and acclaimed
- In use for over 3 years
- No computer knowledge required
- Line printer optional
- Guaranteed

IBM-PC	TRS-80 I/III	APPLE II
$36.95	$34.95	$36.95

- Brain CompuTeasers IBM-PC, APPLE $19.95
- Testmaster Series IBM-PC $19.95 each
- Boggel Word Game IBM-PC $10.95
- Jeopardi TRS-80 I/III $9.95
- Factor Label (CAI) TRS 80 I/III $29.95
- Grade Point Average IBM, TRS-80 $7.95

DEC COMPUTING

Dept. 2
609 Oakleaf Drive • Garrett, Indiana 46738

SAVE MORE THAN EVER ON 3M Scotch DISKETTES!

$1.95 ea. Qty. 20 — 5¼" SSDD (744)

$2.70 ea. Qty. 20 — 5¼" DSDD (745)

5¼" SSQD—96TPI (746) .... $2.89 ea.
5¼" DSQD—96TPI (747) .... $3.95 ea.
(Specify soft, 10 or 16 sector.)

8" SSSD (740) .... $2.07 ea.
8" SSDD (741) .... $2.54 ea.
8" DSDD (743) .... $3.30 ea.
(Specify soft or 32 sector.)

Save even more on quantity orders!

Minimum Order: 20 diskettes. Add $3.00 shipping per 200 diskettes. C.O.D. charge $1.65 additional.

Visa • Mastercard • Checks

For fast service, call
DISK WORLD!
Nationwide: 1-800-621-6827
In Illinois: 312-944-2788
DISK WORLD!

348

Suite 4806 • 30 East Huron Street • Chicago, Illinois 60611

Authorized Distributor
Information Processing Products
3M

The **Drawing Board**

- Circles, Lines, Paintbrush, Text, Animation
- Fill, Mirror Image, Inverted Screen, Enlarge
- Saves graphics as BASIC program
- Screen dump to printer - full or half size
- Disk or tape storage
- Model I, III, 4
- Documentation included

Tape $24.99 Disk $29.99 Manual only $5.00*

Send Check or Money Order (indicate model)
J & A Associates
P.O. Box 1894
Wheaton Station, Maryland 20902

* Purchase price may be applied to tape or disk at later date
Maryland residents add 5% sales tax.

437

DATA COMPACTOR

Increase data storage capacity 1/3 to 100% without using machine-code!

These routines are written in BASIC SOURCE CODE which may be integrated into user's programs in direct sequence or as subroutines.

This manual will show you how to do it yourself.

Send $49.95 & $2.00 USA shipping

Demo disks available with mergeable routines for CP/M or TRSDOS compatible routines for TRS-80, II, or III.

Specify CPM or TRSDOS & add $19.95 for 5¼" disk or $24.95 for 8" disk.

VISA **Dealers Invited**

EDWARDS & ASSOCIATES
P.O. BOX 42158
COLUMBIA, S.C. 29240

435

desktop unit or in a 19-inch rack. Its rugged rotary switch with silver alloy-plated contacts is rated at a 10,000-operation life expectancy. The aluminum enclosure controls EMI/RFI emissions. Each module has an interlock for quick and easy installation. You can piggyback the device to increase computer productivity.

The Mini-T-Switch is available for a risk-free 45-day trial period in versions compatible with either coaxial or EIA RS-232 connectors priced $125 and $230 respectively. For further information, contact Inmac, 2465

DIFFERENT TRACK

Take your frustrations out on your computer with Byte Bat.

Bash It Up!

Every one of us, at one time or another, has felt compelled to smash the video screen on our computer. Some of us (admit it) have even done so. . . but the pleasure derived from such actions was short-lived when the bills came in. Now you can beat up your computer to your heart's content with the Byte Bat.

Specifically designed to serve as a frustration shunt, the Byte Bat features a number of digital interface modes, plus BAUD rates (Basic Aggressive Units of Dissatisfaction) from one to 12,675,432. The device is compatible with all computers and operating systems, making it the first universally compatible foamware.

Each kit comes complete with one Byte Bat, a poster showing the device in use, a button, a user's manual, and a warning decal that alerts all who approach that "This computer-friendly liveware is protected by Byte Bat." The Byte Bat is sold for $9.95 from your local computer dealer. If they are out of Byte Bats you can order your own for $12.50 from MicroTie Systems Corp., P.O. Box 8112, Walnut Creek, CA 94546, 800-227-3900, 800-632-2122 (CA).

Reader Service ✓ 574

New product listings are based on information supplied in manufacturers' press releases. 80 Micro *has not tested or reviewed these products and cannot guarantee any claims.*

Augustine Drive, Santa Clara, CA 95051, 408-727-1970.

Reader Service ✓ 571

3-D Graphics

Broaden your imagination by creating three-dimensional views on your Model III with Surface Plot. The program lets you enter an equation of the form Z = F(X,Y) where Z is the height above the surface for a given X,Y coordinate (e.g., entering the equation Z = 10 − X*X draws a hill). You can view the final picture from any position in space so you can see an image from underneath, above, or even inside a hill or valley on the plot surface. You can also specify the size of the resulting image.

Surface Plot automatically removes hidden lines for best results. You can save finished plots on disk or print them on any of 20 popular printers. The program requires the use of Grafyx Solution in order to work (Grafyx Solution is a plug-in, clip-on board that gives you 98,304 points in a 512 by 192 matrix).

The program, 12 hi-res pictures, and manual cost $39.95 from Micro-Labs Inc., 902 Pinecrest St., Richardson, TX 75080, 214-235-0915.

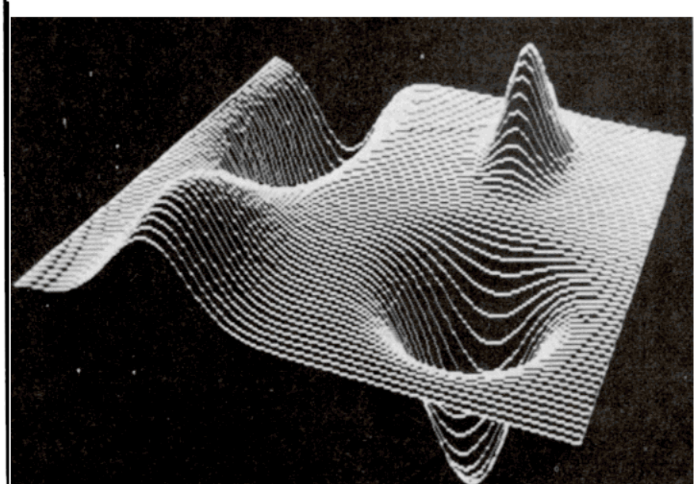
Reader Service ✓ 557

Getting Through the Paperwork

Use Paperwork, a word-processing package, to get through your paper work quickly and easily. The package consists of a Line/Screen Editor that offers global replacement, searches, text line movement, and line insertions without having to position the cursor within the desired lines; a Mailing List Address data base that lets you merge name and address information into letters, envelopes, and mailing labels; and a System Output capability that provides you with hard copies of letters, documents, envelopes, labels, address data base lists, phone lists, and text file copies with assigned line numbers.

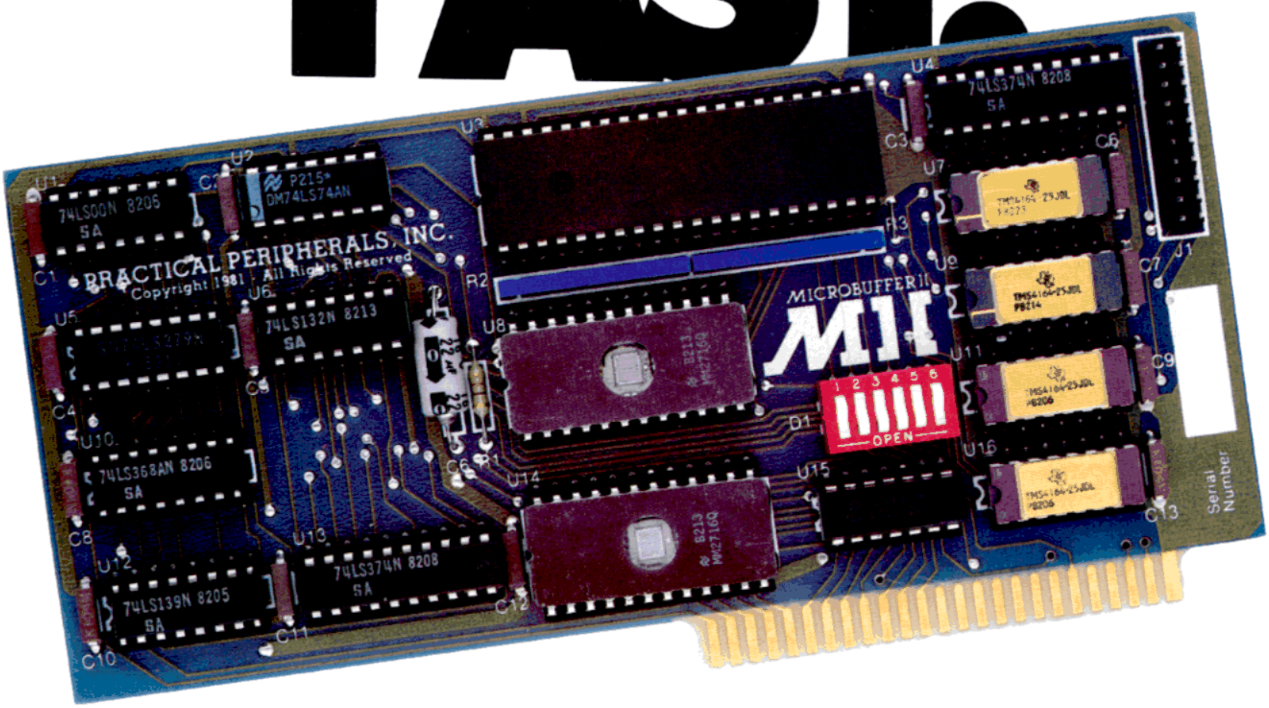
This well-rounded word-processing system is available for the Models II, 12, and 16 operating under CP/M. It requires 64K of RAM and costs just $69.95 before Christmas, $99.95 afterwards. For more information, contact Harris Micro Computers Inc., 2560 North 560 East St., Provo, UT 84604, 801-373-1605.

Reader Service ✓ 580



An example of what Surface Plot from Micro-Labs Inc. can do for you.

LOSE WAIT FAST.™



Don't waste another second waiting for your printer to finish before you can use your computer again. With Microbuffer™ printer buffers you can print and process simultaneously!

MICROBUFFER.™
SO WHAT ARE YOU WAITING FOR?™

Another fine product from **PRACTICAL PERIPHERALS**

31245 La Baya Drive, Westlake Village, California 91362
(213) 991-8200 • TWX 910-336-5431

That's what you get with the LNW80 Model 2—undoubtedly the most versatile, powerful and fully equipped microcomputer in its class today. A machine so superior in concept and design, that it will define the standards of microcomputer performance for years to come.

VERSATILITY

The LNW80 2 performs wonders with the most complete library of software available to any microcomputer on the market today. Every LNW80 2 comes complete with this outstanding library of Business Software. **LNW SMALL BUSINESS AND PROFESSIONAL ACCOUNTING SERIES™** General Ledger, Accounts Receivable, Accounts Payable, Payroll; **ELECTRIC SPREADSHEET™; ELECTRIC PENCIL™ WORDPROCESSOR; MICROTERM™ MODEM PROGRAM; CHART EX™ HIGH RESOLUTION BUSINESS GRAPHICS CHARTING PROGRAM; CP/M 2.2®; DOSPLUS™; LNW-BASIC™; MICROSOFT BASIC.™** In addition to a comprehensive line of LNW80 2 Software, it is also fully compatible with software from TRS80® (Models 1, 3, 4), CP/M® and Cromemco® worlds—a capability which gives you access to the most extensive and mature libraries of business, scientific, engineering and entertainment software applications. So no matter how far you expand into user applications, the LNW80 2 will expand right along with you.

POWER

The LNW80 2 performs miracles with the computing power of 96K RAM (standard) of user memory matched with a mass storage capability which handles 5¼" floppy disks and 5¼" hard disk drives. And while the unit comes with built-in controllers for 5¼" and 8" floppy disks (single/double sided, single/double density, up to 4.5 Megabytes capacity), the LNW80 2 also gives you the unique ability to read and write diskettes from a greater variety of other popular computers than does any other microcomputer. So regardless of how big you grow, you will never end up with thumb-twiddling down time while you expand to a more powerful system. The LNW80 2 will always have enough muscle to handle your biggest and toughest jobs.

FULLY EQUIPPED

The LNW80 2 was developed to anticipate the needs of both expansion and compatibility. So the computer was designed with enough built-in features to keep you from having to spend a small fortune as you move down the road to higher levels of user sophistication. Standard features include high and low resolution graphics in both color and black-and-white, an asynchronous serial communication channel, and a wide variety of tape, printer, monitor and hardware expansion ports. In addition, the LNW80 2 contains an array of quality construction features that fully justify its remarkable one-year limited warranty.

So if you're looking for a microcomputer that will satisfy your performance needs as you grow and develop, take a long, hard look at the LNW80 2. It's the one microcomputer built to meet the challenges of tomorrow—for a long time to come.

For more information and the name of the dealer nearest you write or telephone:

LNW Computers 31
2620 Walnut, Tustin, California 92680
Telephone: 714/544-5744

STATE-OF-THE-ART ENGINEERING STATE OF TOMORROW PERFORMANCE.

LNW COMPUTERS