$5.95 U.S.

two/sixteen™ magazine

+ TWELVE

Volume 1, Number 4 January-February 1983

The journal for business, professional, and scientific members of the TRS-80 community

> "He that publishes a book runs a very great hazzard— since nothing can be more impossible than to compose one that may secure the approbation of every reader."
> Cervantes
> DON QUIXOTE

CONTENTS

the easiest database management system

LUCID

Do-it-yourself mastery within minutes...without a manual

The easiest-to-use system of all, designed by an expert in human-computer interaction. LUCID turns your TRS-80* into a powerful database management system (DBMS), with all the tutorial you need built right into the system. You need no written documentation at all—not a manual, not even a scrap of paper to jog your memory!

Just minutes from start-up to total mastery—guaranteed. Users are comfortable with LUCID in a matter of minutes. That's been proven again and again in practical field testing and everyday use.

Within minutes, you too will use LUCID to sort files...generate reports...enter and change information...screen out selected records...and more. It's guaranteed—or just return the system and get all your money back.

The only database management system to combine all these valuable features in one genuinely easy-to-use package:

- Clear displays and assistance that use capabilities unique to the Model II*
- Meaningful reinforcement every step of the way, for the novice and experienced user alike
- Consistent, logical, and clearly interpreted input techniques, with integrated word processing
- Total isolation from the operating system
- Simplified description of complex retrieval conditions
- Complete error contingencies detect every possible problem ...and recover automatically or suggest how to proceed
- Ability to interrupt or restart procedures, without starting all over again from step one

Look for more details about these and other features in future issues

Introducing LUCID: the cure for those "I-thought-it-was-supposed-to-be-easy-to-use" blues

Menu-driven assistance driving you crazy? LUCID's truly helpful "HELP" key brings you only the guidance you need—and explains any errors you make—at the touch of a button. Instructions are always geared to your specific problem.

Are "English-like" commands leaving you at a loss for words? You could be spending too much time with those supposedly jargon-free programs. LUCID talks your language. So you don't need a programmer or special training to communicate with the system.

Keep getting NO for an answer, when you give a command? LUCID is ready and willing to do it your way, in your style. For example, if your fingers just naturally type "change" instead of "CHANGE," you still get the action you want...not an error message.

Has your "user-friendly" system left you feeling hostile? LUCID applies the latest research in human-computer interaction—and a lot of common sense—to keep you calm and relaxed as you work. You may even find yourself smiling as you breeze through your workload.

If you suffer from any of the symptoms above...

LUCID is the system you've been waiting for. LUCID is designed for *you*, whether you're looking for a database management system to please yourself ...your employees...your customers...or your clientele.

Just install LUCID and start managing records more efficiently, the same day. No time-consuming demonstrations needed before the sale. And no costly courses, or special texts, or training tapes, or utility programs required to support the system afterward.

Yes, LUCID saves valuable time and effort for users, consultants, and retailers alike. So everyone can spend more time getting the job done.

And if you think you've heard it all before... Sample real simplicity with our low-cost demonstration disk. Or send for the whole satisfaction-guaranteed system. Either way, you'll find out for yourself what easy-to-use really means, in a matter of minutes. Guaranteed!

LUCID is available exclusively from

Softerware, Inc. • 200 Office Center • Fort Washington, PA 19034

Call 1-215-628-0400 for fastest service...or return the Coupon below

□ Send me the complete LUCID system for $238.00. If I am not completely satisfied, I will return the disk within 30 days for a full refund.

OR

□ Send me a demonstration disk of the LUCID program for $25.00. I will get full credit for my payment, if I decide to purchase the complete system.

I enclose my □ CHECK □ MONEY ORDER made payable to "Softerware, Inc."

NAME______________________________

ADDRESS____________________________

CITY______________STATE_________ZIP______

TELEPHONE NO. (_____)__________________

Softerware, Inc. • 200 Office Center • Fort Washington, PA 19034 **(215) 628-0400**

*Reg. Trademark Tandy Corp.

two/sixteen
/magazine

Editor In Chief/Publisher	Richard H. Young
Managing Editor	Barbara S. Albert
Art Director	Timothy P. Schleif
Administrative Assistant	Fay L. Smith
Contributing Authors	Arnold Fischthal John Thornton Roger Conant Joe Lyddon Robert VanNatta Jim Kloosterman
Typesetting by	Innovative Ink Lancaster, PA
Printed by	Innovative Ink Lancaster, PA

two/sixteen is published bi-monthly by

two/sixteen

131 East Orange Street, Lancaster, PA 17602

Telephones

Editorial Staff	(717) 397-3365
Classified Advertising	(717) 397-3789
Display Advertising	(717) 397-3364
Subscriptions	(717) 397-3364

Annual Subscriptions:

Third Class	$30 in U.S.
First Class	$36 in U.S.

VISA and MasterCard Accepted

Foreign Subscriptions:

Canada and Mexico	$36 U.S.
Other Foreign	$47 U.S.
Air Mail Foreign	$67 U.S.

Drawn on U.S. Banks

Send all subscription correspondence,
including change of address (with label) to
two/sixteen
Box 1216, Lancaster, PA 17603

two/sixteen welcomes letters to the editor. All letters sent to the magazine become the property of two/sixteen magazine and are subject to publication unless otherwise requested by the author.

EDITORIAL

Operating Systems

Coming from an interactive large-machine background (many years using a DEC-10 with a powerful FORTRAN and System 1022, a superb data base management system, for moderately large volume financial applications), I was fairly quick to move up from the essentially awkward and cumbersome drawbacks of TRSDOS 2.0 to the relatively clean, uncluttered, user-friendly (in the classic sense), faster, and forgiving world of CP/M 2.2 (we use Pickles & Trout) for all of our own work. Virtually all others of similar background that I have talked with have done the same.

One of the reasons we like CP/M is its similarity to the more sophisticated systems we are accustomed to, just as many computer neophytes tend to prefer their first operating system to anything else, regardless of relative features. However, the overwhelming appeal of CP/M, UNIX, p-code, OASIS, and possibly others is that they are designed to run on many different microcomputers, some of which were intended to be repugnant to "foreign" operating systems. For example, we were able to run the Model II version (unmodified) of Pickles & Trout CP/M on the Model 16 as soon as it arrived *without any hitches*! And now, we run the Model 16 double-sided-disk version of CP/M in both the II and 16 without any modifications. In the long run (meaning a year or two), general acceptance of "mainstream" operating systems will be so great that no parochial systems will be able to survive into subsequent generations of computers. This will mean that software developed under such limited systems will then be usable on only technologically obsolete hardware (though, of course, such hardware may be physically useful for many years to come).

So, whether anyone likes it or not, we'll all be converting to one or another of a few "standard" operating systems before very long. And such a conversion should be complete before making any significant investment of time and money in software development, or the investment may be prematurely lost.

There are many rumors floating around to the general effect that Radio Shack will be moving into "mainstream" operating systems (such as CP/M) for its serious computers (the II, 16, and the new 12, which is scheduled to be announced by the time you read this). What seem to be very reliable rumors have a primitive multi-user system (probably another Ryan-McFarland product) being offered free by Tandy to Model 16 owners by late January, and a full UNIX-like system (Microsoft's XENIX, after all!) on February 15 at a cost of $750, including a C compiler and a mail system. Rumor has it that this full-scale XENIX will require 256K memory and a hard disk. Other expected 16-related multi-tasking product announcements are a *compatible* interpretive BASIC at $300, to be followed with a COBOL compiler at $699, Multi-plan at $350, FORTRAN, and PASCAL. Seven business packages are expected to be out in 16 versions by the end of February. Whether these rumors are true or not, I'm convinced that Tandy really will begin to support standard operating systems, since they must want to retain a significant share of the business market. Large companies are certain to be major users of serious business micros, and it is almost inconceivable that a large company would put hundreds of computers on local networks and run them under TRSDOS. The options are far too limited.

This issue contains a three-way comparison between CP/M systems (Pickles & Trout, Aton, and Lifeboat). TRSDOSers will readily see that all is not perfect in CP/M heaven, but should recognize that there are, in the final analysis, just three paramount considerations: (1)***standardize***, (2)***standardize***, (3)***standardize***. All other considerations are relatively unimportant.

New publications

We are planning two new publications. The first is a newsletter which we intend to introduce in February or March. It will provide up-to-date information for corporate chief executives on the state of microcomputing hardware and software development, and will also provide reasoned speculation on its short-term (six months) and long-term (two years) prospects. We are negotiating for the services of one of the country's most experienced and successful magazine editors to assist us with this publication. Provisionally titled *Microcomputing Early*

Warning Letter, it will be published semimonthly at an annual subscription rate of $250.

The second is a monthly magazine, *MainStream Microcomputing*, which will debut in the spring. It will provide information to help microcomputer users get into the "mainstream" of common operating systems, languages, and peripherals, and to help them remain in the best position to take advantage the new hardware and software developments with a minimum of intergeneration conversion cost. Many of us believe that a new "generation" of microcomputers will be introduced every year for several years (perhaps culminating in 64-bit computer chips, multi-billion byte solid-state non-volatile memory, and flat screen, all in a computer the size of an 8-1/2"x11" ring binder when folded shut). Subscriptions to *MainStream* will be $36 per year.

Subscribers to *two/sixteen* may subscribe to these new publications at the special discount rates of $170 and $24, respectively, until March 1, 1983.

Other new business

Good furniture for microcomputers and terminals has always been hard to find. Most of the desks or terminal stations have sharp corners. Much of the furniture is not very sturdy. Ordinary desks are uncomfortably high.

We've recently discovered a line of furniture that meets our specifications, and we've standardized on it. It's the Data-Mate line by Maine Manufacturing Company. We like it so much that we've decided to sell it both here in Lancaster and to any subscriber who needs really good computer furniture at a reasonable price. You don't have to wait weeks for this furniture, as it will be shipped from the factory (in New Hampshire) within four days of order. We can't afford to carry large receivables, so we are giving a 15% discount for cash with order. See our ad in this issue for more details. We'll be happy to send you a catalog with more information.

Games for the Model II and 16 have always been hard to find, and many users would like something to relax with at the end of the day or to show to their non-computer oriented friends. We've been selling STARSHIP I for over a year, and its owners report great satisfaction with it. It's got some graphics, but its mostly strategy, albeit against a clock. Many users have expressed regret that there were no arcade-type games for the II or 16. I'm pleased to announce that this problem is now overcome.

We are selling a set of five arcade-type games for the II and 16. They are written in RS interpretive BASIC without any fancy poking, yet they are faster than anything else like it that I have seen. These are products of a new venture, "Maryland Model II Games," and the set sells for $29.95. This is MM II Games first venture into game sales, but I understand that if these achieve acceptance there are more waiting in the wings. From my viewpoint, these games range in difficulty from fairly difficult to nearly impossible. However, I've seen a high-school student appear to master them all. Send us $29.95 (or we'll take your phone order on VISA or MasterCard), and we'll send you a disk and documentation postpaid.

New products received

Two new accessory products for II and 16 micros have been received recently. The first is an orange video tube produced by Langley-St. Clair. It can be installed with basic tools and comes with complete and clear installation instructions. It takes about an hour to install, and perhaps one to three hours more to adjust (allowing time for high voltage bleed-off).

The main virtue of this tube is that it is flicker-free, but many people also prefer the esthetics of the orange tube. I personally prefer the orange tube to the white tube that came with the II, but I much prefer the resolution of the 16, flicker and all, to the II with the orange tube.

We've heard from one subscriber who installed a Langley-St. Clair tube in a 16 and likes the flicker-free result much better. He says that the resolution is also just as good as ever.

The decision to buy such a tube will be based on very personal preferences rather than any sweeping logic that applies to everyone. The orange phosphor tube costs $89.95 (a green version is available at $79.95, and anti-glare versions of each are available for $10 more). If the flicker gives you eye fatigue in long sessions at your micro, one of these tubes is probably worth the price. Some will find the orange or green colors more soothing or just more esthetically pleasing.

To order or get more information contact Langley-St. Clair Instrumentation Systems, Inc., 132 West 24th Street, New York, N. Y. 10011, (800) 221-7070.

Boot Error Eliminator

The second new product is a Systems Enhancement Engineering Boot Error Eliminator (SeeBee). This is strictly for Model IIs which require terminator plugs when an expansion bay is not plugged in (generally, those purchased before about April, 1982).

The problem with these earlier IIs is that if you boot the system with the expansion drive(s) turned off, the disk in drive 0 (or A if CP/M) is usually severely damaged. With the SeeBee unit installed, the expansion unit may be left off during all single-drive operations. It can be turned on or off anytime without damage to the drive 0 disk. (Of course, you shouldn't turn any drive off or on while it has a disk in it, even though the probability of damage to the disk is slight.) The prototype model we have works perfectly.

The SeeBee is a tiny unit that plugs into the disk expansion port of the II. The expansion unit ribbon cable is plugged into the SeeBee.

I think this is a much better solution to the problem than the $69.95 Radio Shack Automatic Control Power Strip (26-1429) listed on page 42 of Microcomputer Catalog No. RSC-8. Using this power strip, it is still easily possible to boot with the expansion bay off and damage your drive 0 disk. Another alternative (but expensive) solution is to have Radio Shack modify your II so that it meets the newer specs. There is apparently no standard price for this mod, as I've heard estimates of from $120 to $300 from various Computer Centers.

In my opinion, all users of older Model IIs, especially those who must use clerical personnel to boot the machine, should get a SeeBee as cheap insurance against the dreaded boot error.

The SeeBee costs $59.95 plus $3.00 shipping from See, Inc., P.O. Box 40215, Indianapolis, IN 40215, (317) 844-8817.

If you order one of these products, please let the seller know where you heard about it.

New Service For 9-Track Tape

We've just learned of a firm which will convert from tape to diskette or vice-versa. They can handle 9-track 1600 BPI EBCDIC or ASCII tapes with block sizes of 80 to 2048 characters. Diskettes may be for Model II or Model III and may be in either CP/M or TRSDOS format. Tape to tape processing may be done to allow transfer of larger block sizes, and fast turnaround is available. For more information, contact Warren Glimpse & Company, 105A Orinoco Street, Alexandria, VA 22314, (703) 836-6800.

NewsNet

We're not on the wire yet, because we haven't had the time to produce abstracts of our issues. However, we expect to have the current and prior issue on the wire by January 31. All subscribers should have received a 4 page insert with this issue explaining how to get validated for this service. Many subscribers will find other publications on NewsNet which can help them in their business.

SOFTWARE REVUE

REVIEW OF CP/M VARIANTS

LIFEBOAT, PICKLES & TROUT, AND ATON

Robert P. VanNatta
222 South First Street
St. Helens, Oregon 97051

By now, surely everyone knows that CP/M is **the** operating system for Z80 based microcomputers. Yes, even Tandy knows this.

It is no accident that the TRS-80 II/16 line of computers is among the best performing CP/M computers on the market. When the Model II became available in 1980 at $3899, it had the best price-performance ratio on the market. Times have changed a bit in the last three years, and now, of course, it looks a bit overpriced even at its recently advertised price of $2999. The point of this article, however, is not to berate Tandy for failing to cut their prices fast enough to keep up with the competition but rather to discuss an outstanding feature of the Model II/16 that Tandy refuses to discuss publicly, i.e. the ability of the II/16 to run CP/M.

CP/M Is An Operating System

Rumor has it that about half of the Model IIs produced are "inflicted with" CP/M. Contrary to other rumors (no doubt originating in some Texas cowtown), CP/M is not a disease that is spread by the California fruitfly. Back in the dark ages, before CP/M became a word in the English language (and a trademark of Digital Research), the letters were understood to stand for "Control Program/Module." Actually, CP/M is an operating system dating back to the days when Tandy was selling horse blankets and CB radios (circa 1974).

The concept of exactly what an operating system is is not the easiest thing to grasp, particularly if you are not a programmer. Stated crudely, the operating system is the program that gets you from the point where the computer says "Insert Diskette", to the point where you are ready to load an applications program. (This point is identified by A> under CP/M and the words TRSDOS READY on Tandy operating systems.) Stated technically, an operating system is the interface between the hardware and the applications program. Stated figuratively, an operating system is to a computer as a steering wheel and gas pedal are to an automobile.

The beauty of CP/M is not so much in its brilliant organization but rather in its standardization. After all, the first concern with a plug is that it fit the socket. Questions of whether it is brilliant or not are of strictly secondary concern. A misunderstood concept surrounding CP/M is that it is not as standard as everyone assumes. Much has been written about the "standard face" of CP/M. This is the face that lets you load and run a variety of programs that may have been written on one brand of computer to run on quite a different computer.

Obviously, all computers that run CP/M are not the same, and if they are made to look the same, someone has to do some fancy footwork. This fancy footwork is called the BIOS. The BIOS is the Basic Input/Output System, which is, by definition, hardware-dependent. "Hardware-dependent" is a fifty cent word that simply means the BIOS must be custom written for every brand of computer. Machine-specific code to support the disk drives and printer ports are examples of items in the BIOS.

CP/M Must Be Customized To Each Computer

The next myth associated with CP/M that needs some attention concerns the actual product sold by Digital Research. It is pretty well understood that Digital Research is the source of CP/M. What is not understood is that Digital Research only markets the kernel of CP/M; it does not ordinarily furnish the BIOS. The responsibility for writing the code to hook up CP/M to a particular computer is the responsibility of some third party. In most cases, the hardware manufacturer will either write the BIOS, or else he will contract with a software house to write the BIOS for his computer. The result is then marketed with the computer as its operating system. This arrangement normally works quite well as the hardware manufacturer has control over the BIOS and can assure that the hardware features are implemented in a reasonable fashion and can, if necessary, make appropriate hardware adjustments.

In this sense, the situation with the Model II/16 is quite unusual. Since Tandy refuses to publicly acknowledge the existence of CP/M, they have obviously not undertaken the responsibility for writing a BIOS to implement it.

BIOS Gives Computers Their Personality

Tandy's cop-out has resulted in several independent software houses writing BIOS for the II/16 computers. Since this work has necessarily been done without the help or even cooperation of Tandy, the BIOS writers have had to guess at the features that ought to be implemented, and have likewise had to guess at the best way to get from here to there. The differences in the implementations have necessarily influenced the ultimate personality of the computer, with the result that the II/16 tends to be a schizophrenic computer. A Model II/16 running under Lifeboat CP/M is quite a different computer from one running under Pickles and Trout CP/M. It is different still if it is running under ATON CP/M.

The Versions

The goal of this article is to discuss these personality variations. Pickles and Trout CP/M and Lifeboat CP/M have both been around since early 1980. ATON CP/M gained attention in 1982. I first acquired Lifeboat CP/M in about March of 1980 and have used four or five different versions intensively since that time. I have had Pickles and Trout CP/M on hand for about six months, and a review copy of ATON CP/M was relied upon to compile the information presented here. The versions reviewed are Lifeboat 2.25d, Pickles and Trout version 2.2eD, and ATON 2.23HE.

I own several Model IIs and a Model 16. This article is being written on the Model 16, expanded with a homebrew expansion chassis originally assembled for a Model II consisting of three Shugart 801 single sided drives (the kind used in the Model II terminal). I modified the expansion chassis to work with the Model 16 by the simple expedient of pulling the data cable off the back of drive B (in the expansion box), leaving drives A and B in the Model 16, and C and D in the expansion chassis to function.

Lifeboat Is Vanilla?

Obviously, I'm most familiar with Lifeboat CP/M, and if I seem biased in its favor, at least you know why. Over the years I have worked (or contended) with five versions of Lifeboat CP/M. (There were others also.) Some have been more forgettable than others. Version 2.25d and version 2.24 and earlier will work on the Model 16 as well as the Model II. Versions 2.25a, b, and c won't work on the 16 and on some Model IIs. (Tandy changed brands of disk drives without telling anyone.) Of all the variants of CP/M for the II/16, Lifeboat CP/M would have to be labeled "plain vanilla." Its documentation consists of a scant 24 pages, together with the infamous seven Digital Research booklets. A pleasant CONFIG utility is provided to set the I/Obyte and to configure the printer ports for protocol.

What's An I/Obyte?

For those of you not familiar with CP/M, I probably should take time to explain the I/Obyte. CP/M is structured in such a way that there are four logical input/output connections called logical devices. They are: (1) CON for console, (2) RDR for paper tape reader, (3) PUN for paper tape punch, and (4) LST for list device. (The names of these devices give a hint as to the age of CP/M.) Each logical device may in turn be connected to one of several physical devices. Normally the CON(console) is set to the keyboard, and the CRT and the LST(list) are set to the printer. The I/Obyte is a single byte stored within CP/M that is accessible (primarily to machine language programmers, but also by use of a POKE from BASIC) to control the assignment of the logical devices to the physical devices. CP/M easily supports both serial ports and the parallel port on the II/16 by simply redirecting the LST output to the appropriate physical device. (The CP/M Manual documents the use of STAT.COM to accomplish this redirection, however all three versions of CP/M reviewed provide a thoughtful menu driven utility to simplify the task.)

For example: If you so desire, you can easily redirect the printer output right back to the screen by setting the list device equal to the CRT. In a similar vein, you can actually disconnect the keyboard and screen from CP/M by redirecting the console logical device to a serial port. (Don't ask me how to issue the command to reconnect.) The point is that this feature permits you to switch off the usual terminal and connect a remote terminal as the input device. (I haven't done it, so all I can say is that this is conceptually true; actual mileage may vary.)

All three CP/M versions support a wide range of baud rates on the serial ports and support both software and hardware handshaking conventions. ATON and Pickles & Trout permit word size and parity configuration, as well. Lifeboat does not.

All three versions furnish a convenient menu-driven configuration program to set up the the printer ports and various other features. Once set up, these features are a permanent part of CP/M and will remain as set until you rerun the setup program. All versions also allow temporary alterations in the configuration.

Drive Support

Lifeboat has the weakest disk drive support of the three; it is the only one that does not support double-sided floppy disk operations. What this means to you as a Model 16 + Lifeboat CP/M user is that either single or double-sided diskettes in the double-sided drives can be used, but all you will get is data on one side of the diskette and a drive capacity of 596k.

Among these versions of CP/M, three data densities and four data formats are available. All support the standard 128 byte per record, 26 records per track density. This is the time-honored 8-inch single density.

For those new to this business, I should observe that almost every CP/M computer with 8-inch drives ever built will read and write to this single density (241k per disk) format, and this format provides the standard medium for exchange of data among computers. For example, much of the mailorder computer program business is done in this format. By contrast, there is no such thing as a universal 5-inch format.

The Radio Shack double density format (generated with 256 byte records, 26 per track) is supported by both Lifeboat and ATON but not, however, by P & T. This format yields approximately 485k per single-sided drive, meaning that the FORMAT and COPY utilities of either Lifeboat or ATON CP/M may be used to duplicate a TRSDOS II diskette. This does NOT mean that CP/M can read the TRSDOS Diskette. It can't — without a special utility — because the directories are not compatible. It does mean that both ATON and Lifeboat provide a bit-copy program that will simply clone a disk without knowing its contents. Incidentally, the CP/M copy procedure is MUCH faster than the TRSDOS procedure. (This should not be confused with the P & T and ATON utilities which permit transfer of a TRSDOS ASCII data file from a TRSDOS disk to a CP/M disk.)

The density you will actually use is a density that yields about 600k per single sided drive. Lifeboat calls it "extended" density and obtains this capacity using the IBM double density format of 1024 byte records placed eight to the track. ATON does likewise but calls it double density. By contrast, P & T obtains the capacity with sixteen 512 byte records per track and calls it double density.

As you might guess, P & T and Lifeboat CP/M cannot read each other's disks, except in the single-density interchange format.

This is one place where ATON is ahead of the pack. ATON works on the Lifeboat format of 1024 bytes per sector, but it will read and write to single-sided P & T 512 byte disks. This means that if you boot on ATON CP/M, you can mix and match P & T and Lifeboat diskettes in your other drives and move files between them without difficulty. Two points should be made, however. First, the use of P & T formatted disks under ATON causes a considerable loss of performance and ought not to be done, except for the purpose of transferring files. Second, ATON cannot handle a two-sided P & T disk.

All CP/M variants will identify the various types of disks supported, and the drives will configure themselves to the appropriate format without user intervention (except for a warm boot).

Error Trapping

Lifeboat CP/M follows the CP/M conventions with respect to system in disgusting detail. For the most part, operating system errors are not trapped, and often, error messages are not returned. For example: If you attempt to select a drive that is not ready for some reason such as (a) it doesn't exist, or (b) the door is open, Lifeboat CP/M will hang without comment and wait patiently for you to figure out what the problem is by yourself. Pickles & Trout CP/M will print a message on the bottom of the screen and then hang. ATON will blink the entire screen at you and print a message on the top of the screen telling you what is wrong. If, for some reason, you are unable to bring that drive on line, you then have the option of redirecting the disk access physical drive A by forcing the system into the one-drive mode. The only problem with the ATON approach I found is that once you are in the one-drive mode, I could not figure out how to get out of it without a cold boot.

One-Drive Support

All three versions of CP/M provide credible support for single drive machines; they all reassign the single physical drive to alternative logical drives. Lifeboat provides a very convenient FILECOPY utility that reads a file into memory and then prompts for a disk change, without playing the logical drive game. My mind sometimes goes wacko in response to some inane, albeit logical, command such as "Insert Disk B into drive A".

ATON has a utility aimed at the hard disk market that permits a file larger than the destination disk to be broken up during the move and placed on several destination disks if necessary. I don't have a hard disk so I couldn't test it, but if it works, it would relieve a serious concern I have about using hard disks, i.e. backing up files larger than a floppy.

Terminal Features

Lifeboat supports terminal control codes of the Lear-Sigler ADM 3A and ADM 31 terminals. Pickles & Trout has its own unique terminal control codes. They are not even similar to anything you have seen before. ATON ships its version of CP/M compatible with the ADM 3A and the SOROC 120 terminals. ATON, however, provides a menu-driven user-accessible jump table in its configuration routine that allows you to reset the terminal control codes to anything you might desire. As nearly as I can tell, the difference between the SOROC and the ADM 31 codes is only the ESCAPE sequence to clear the screen.

For what it's worth, ADM 3A compatibility is recognized as a good feature, not because the ADM 3A was a great terminal, but rather for the same reason that plumbers put hot-water faucets on the left—standardization.

ATON implements the bell function (Ctrl G) by blinking the screen at you. Unlike its competitors, ATON optionally supports a 25 by 80 screen format. This is accomplished by squeezing the regular 24 lines together slightly and showing a status line at the top of the screen. In this mode, the time and date are continuously displayed. Obviously, if you want the correct date or time you must either set it after you boot up or have a Pickles & Trout TRISWATCH. ATON and P & T both document methods to directly access the memory mapped video board and to define the personality of the cursor. Lifeboat does not. As shall be mentioned later, WordStar can be installed to utilize the video board on Lifeboat and P & T CP/M but not on ATON CP/M.

The Keyboard

Under Lifeboat CP/M, all keys generate the output described in the Radio Shack Manual. P & T almost followed this convention except the HOLD key was changed so it outputs a Control S. ATON went hog wild on the matter. It furnishes a jump table in its SYSDEF program that allows you to define almost any key to mean almost anything. If the idea of having a "b" appear on the screen when an "a" is typed turns you on, you will love ATON CP/M. ATON is shipped with quite a few keys revised; however, I am not sure these revision have improved things much. Fortunately, it wasn't too hard to find the jump table and change the keys back to their conventional values. If you have never used CP/M before on this computer, I suppose ATON's revisions wouldn't be as annoying as I found them.

One thing Radio Shack did wrong on the Model II/16 was they forgot to put a key on the keyboard to implement the accent grave (ASCII char 96). This oversight on the part of Radio Shack has frustrated me for years, as my NEC Spinwriter hides the mathematical symbol for the degree sign behind this code on many of their thimbles, thereby rendering it inaccessible to me. ATON solves this problem by allowing you to assign almost any key that value. Tandy must have saved at least fifty cents by leaving the ASCII character 96 off the keyboard.

Documentation

Documentation is a touchy subject with computer programs anywhere. The diverse approaches are interesting, however!

Lifeboat implements almost no special features over and above those documented by Digital Research, and, accordingly, its special documentation could be generously described as spartan.

Pickles & Trout has extensive documentation that is quite informative. It even draws a picture explaining how to construct a null modem so you can make your serial printer work on a II/16. Unfortunately, this manual has been updated several times without reprinting, and it is beginning to look like something designed by a committee.

The ATON CP/M documentation has really impressed me with its organization and clarity. It is still fresh (dated September 1, 1982), so I suppose the addenda will eventually mess it up, too. One feature in the ATON documentation that will make me forever appreciative is its explanation of the jumper settings for double-sided drives for the Model II. Pickles & Trout provides a version of CP/M to work on double-sided drives but includes not a hint of how to configure the drives. Lifeboat has, of course, ignored the double-sided issue entirely, at least so far. I suspect, however, that it will reconsider the double-sided issue, when (and if) it implements CP/M 3.0 as I look for Tandy to add double-sided drives as standard equipment on the Model II during 1983.

Hard Disk

Lifeboat and Pickles & Trout have a Corvus package. Pickles & Trout and ATON have versions for the Radio Shack Hard disk. I don't have a hard disk and don't know how well these extensions work. (I never will have a Radio Shack hard disk unless they become a bit more realistic about their price structure.)

Extra Utilities

Lifeboat provides all the standard Digital Research utilities plus an outstanding disk copy program. It also provides a dumb terminal routine.

P & T has changed things around quite a bit. It includes some very good diagnostic routines, for example. Unfortunately, it has mangled some of the Digital Research utilities. Specifically, it merged MOVCP/M and SYSGEN into a single routine called RESIZER. It has further mucked up RESIZER so it will only work from the master distribution diskette. I flatly refused to consider buying an Apple because I didn't want to play that game, and here it is.

The irritation of RESIZER is having to go through the hassle of reconfiguring everything each time you create a new system disk. My habit is to put a system on every disk I format. When you consider that I have 50 to 100 disks in the drawer beside my computer for my day-to-day use, you can imagine what I think of RESIZER.

SYSGEN, as provided by ATON and Lifeboat, also has it problems. In both cases the I/Obyte setting will flop back to the default parallel port. This alone is enough of a nuisance, without the problems deliberately created with RESIZER.

The solution I have found to all of this is to make use of Lifeboat's COPY utility. It contains an option to "copy system tracks only." Since it is a bit-copy routine, it copies Pickles & Trout system tracks just as well as it handles Lifeboat tracks and does so without messing up the configuration. It will not work on two sided disks.

The ATON copy program has no way to limit itself to copying just the system tracks; it insists on copying the entire disk. Curiously, however, if you undertake to clone a P & T two-sided disk with it, it will clone the front side of the disk only. This successfully moves the P & T system, but when you check for your files that were on the back side of the disk, you will find only the directory entries got moved, not the file. Ouch!

Pickles & Trout does not furnish a bit-copy program at all; instead, they provide a utility called FASTCOPY. In reality, FASTCOPY is a work-alike of PIP with improved buffering. As with PIP, this is a file-oriented (as distinguished from a track-oriented) transfer utility. FASTCOPY has the usually favorable side effect of reorganizing the disk and rationalizing the file structure as the backup is being made.

There is a subtle difference between cloning a disk and transferring the files from it to another disk. For the most part, these differences are completely transparent to the user. But if the physical location of a file on the disk is of some relevance to you, you may rest assured that FASTCOPY will rearrange it for you. An advantage of FASTCOPY, or for that matter, just plain PIP, is its ability to permit you to move files onto a disk with some files already on it. It will also work in conjunction with utilities that identify bad sectors and lock them out of use.

A serious problem with a bit-copy program is its very unforgiving nature when it comes to mistakes. If you stupidly transpose the source and destination diskettes, it will dutifully copy all 77 empty tracks over the top of your full ones. What's worse is that it transfers the directory first, since it is one of the first tracks copied. Thus, even if you figure out your mistake before the whole job is done and do something radical like shut off the computer, all is lost.

By contrast, FASTCOPY, which is file oriented, moves files in an orderly manner. It will not overwrite files on the destination disk unless they are of the same name. Even if they are of the same name, it methodically writes out the files on the destination one at a time, so if you recognize the errors in your ways before the job is completed and take radical action to stop the computer, you may actually save something. Radical action includes doing one or all of those things the instruction manual tells you never to do, such as: (1) opening the drive door and jerking the disk out; (2) smashing the computer; (3) turning off the power; or (4) pushing the reset button.

ATON provides a directory display program (S.COM) reminiscent of the old XDIR programs from the CP/M users group. It works, and displays the directory in alphabetical order (on a 5 column format) followed by the file block size.

Bugs, Irritants, And Non-features

One of the most difficult jobs of review writing is the thankless task of disclosing the undesirable features of a product. The problem is that one man's feature is another man's bug. Likewise, personal perspective and prejudice invariably become involved in discussing someone else's faults.

I have never seen a version of CP/M that met the impossible goal of being all things to all people. Of the versions herein reviewed, each has its good points and its weaknesses. Please bear in mind that depending on your particular application, many of these issues may be irrelevant.

Lifeboat

I am most familiar with Lifeboat CP/M, having used it intensively for almost three years. The people at Lifeboat have had their share of frustration getting a version that works and keeping it working while Tandy makes subtle changes in the hardware. Its most glaring weakness is the absence of support for double-sided drives.

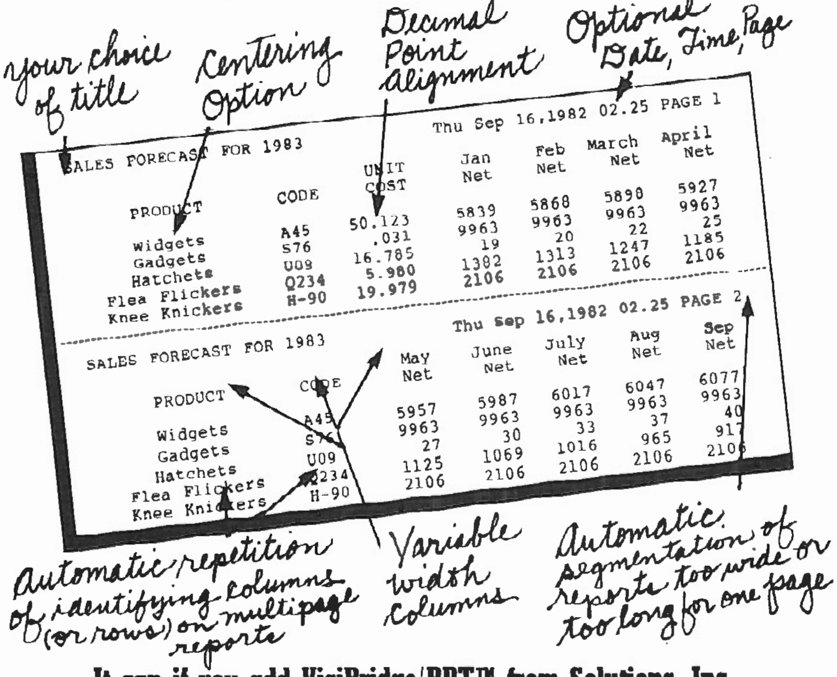
A very minor bug exists in the FORMAT utility relative to the Model 16 in that it fails to let a drive which has timed-out get up to speed before it goes to work. This bug will cause the program to erroneously report a defective disk. You can work around it by immediately restarting the program before the drive has a chance to time-out once again.

Also, Lifeboat has only one switch for adjusting the seek rate. If you have mixed and matched drives as I have, this will require you to set all your drives to the seek rate satisfactory to your weakest one.

If you are a non-programmer who intends to use off-the-shelf CP/M software on single-sided drives, Lifeboat CP/M will serve you well. Commercial software writers invariably attempt to design a program with the broadest possible potential customer base. To accomplish this end, they tend to avoid special function keys and other whistles and bells in order to preserve as much portability as possible. Lifeboat is a big marketer of off-the-shelf CP/M software, and a bit of reflection will possibly tell you why Lifeboat CP/M makes the Model II/16 look like a run-of-the-mill CP/M computer that is solid on reliability and short on exotic features.

WordStar

I use WordStar as my primary word processor. WordStar is acclaimed in the press as being the most popular word processing program on the market. This



```
                                   Thu Sep 16,1982 02.25 PAGE 1
SALES FORECAST FOR 1983
                              UNIT      Jan     Feb    March   April
                  CODE        COST      Net     Net     Net     Net
     PRODUCT                                                    5927
                                        5839    5868    5898    9963
     Widgets       A45       50.123     9963    9963    9963      25
     Gadgets       S76         .031       19      20      22    1185
     Hatchets      U09       16.785     1382    1313    1247    2106
  Flea Flickers    Q234       5.980     2106    2106    2106
  Knee Knickers    H-90      19.979
---------------------------------------------------------------------
                                   Thu Sep 16,1982 02.25 PAGE 2
SALES FORECAST FOR 1983
                                May     June    July    Aug     Sep
                  CODE          Net     Net     Net     Net     Net
     PRODUCT                                                    6077
                                        5987    6017    6047    9963
     Widgets       A45          5957    9963    9963    9963      40
     Gadgets       S76          9963      30      33      37     917
     Hatchets      U09            27    1069    1016     965    2106
  Flea Flickers    Q234         1125    2106    2106    2106
  Knee Knickers    H-90         2106
```

your choice of title
Centering Option
Decimal Point Alignment
Optional Date, Time, Page
Automatic repetition of identifying columns (or rows) on multipage reports
Variable width columns
Automatic segmentation of reports too wide or too long for one page

may very well be true, but WordStar is an intensely complex program. Getting WordStar to run, and getting it to run well, are two quite different issues.

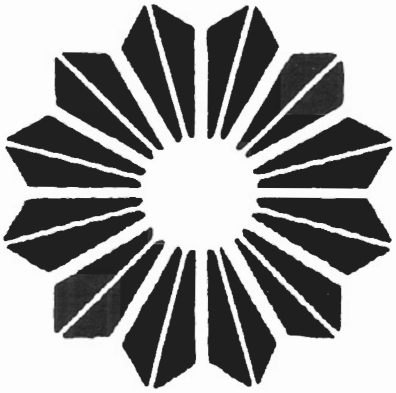
Two features are very important to top performance of WordStar. One is a memory-mapped Video board (for rapid screen revisions), and the other is a keyboard buffer to assure no input is lost during the unpredictable disk accesses and interrupts for which WordStar is so famous. ATON level I flunks on both counts. Both P & T and Lifeboat have menu installation routines built into WordStar that patch the II/16 video board into life. Neither of these will work under ATON CP/M. This forces installation under the much less desirable serial terminal mode. If you have never used WordStar on a video board, you might not be offended by the terminal mode installation, but I can assure you that if you have ever run WordStar on a video board, you will find a serial terminal very annoying, if not downright unsatisfactory.

The lack of a keyboard buffer means you must honor the WordStar "WAIT" light with great care. For example, the standard routine to save and continue editing is Ctrl K Ctrl S Ctrl Q Ctrl P. If you hit those four keys in rapid succession under ATON CP/M, the disk activity generated by the "save" will often cause the Q to be lost. The result is that you end up on the print menu instead of where you expected.

Between Lifeboat and P & T, the quality of the WordStar installation available is very close. There are four reasons for my personal preference of the Lifeboat installation: (1) the P & T implemention does not leave the cursor blinking and the Lifeboat does; (2) Math* version 3.043 (an augmentation of WordStar marketed by Lifeboat that gives WordStar the ability to add, subtract, multiply, and divide on screen) does not work on either video board package; (The absence of a serial terminal installation option makes this program unusable on P & T CP/M.); (3) SpellStar Version 1.2 fails to clear the screen properly with the P & T package. (Version 1.0 wouldn't work at all on either video board package, but would run [sort of] under Lifeboat serial terminal emulation.); and (4) the serial port driver installation option of WordStar doesn't seem to work under P & T CP/M, which means you have to run your serial printer as a CP/M List device. This advantage is only a significant issue if you want to use multiple printers on the same computer.

For example: if you have a parallel printer and a serial printer both installed on your computer, you can generate a version of WordStar by using either Lifeboat or ATON CP/M that will bypass the CP/M List device and drive the serial port directly. Configure your operating system with the LIST device set to the parallel port, and you will generate a second version of WordStar that drives the LIST device. Then you can make both versions coresident on the same disk and control which printer you are going to use by deciding which version of WordStar to load, without ever having to bother with resetting the I/Obyte.

ATON CP/M and WordStar simply do not get along with each other. Superficially, WordStar appears to work on ATON CP/M when installed following the Lifeboat instructions. ATON cautions you not to try it, but I, of course, did anyway. What happens is that shortly after you get started, the system locks up. The clock display stops ticking, and the keyboard becomes completely unresponsive. This lockup seems to occur randomly, but usually within the first minute. I believe the explanation is that WordStar uses a lot of interrupts and so does ATON. I suspect the lockup is the result of these interrupts colliding. Anyhow, I quickly learned that I ought to follow the instructions and use a serial terminal emulation. The result was an improvement but not a cure. I found by using the SOROC terminal emulation that I could go five or ten minutes between lockups. I thought about that for a while and then went into SYSDEF and disabled the status line and turned the clock off. (I sort of missed the clock because when it was turned on, it was a reliable indicator of my fate; when it stopped, I knew exactly what the score was). After I turned the clock off, things improved considerably. I found that my Model 16 would work an hour or so between lockups.



I actually managed to get about three-fourths of this article written using ATON CP/M before I got so frustrated that I went back to something that worked.

None of the possible alternative menu installations make the arrow keys do anything useful in WordStar. The ability of ATON CP/M to assign alternate values to keys means that you can, by redefining these keys, assign them any WordStar single keystroke command. A more reasonable approach, however, is to add some simple patches to WordStar to make the keys do whatever you desire. A detailed tutorial on how to do this was published in the April 1982, issue of *Lifelines, The Software Magazine* which is published by Lifeboat Associates.

ATON

It might be tempting at this point to write off ATON CP/M as a dog and forget it. I wouldn't do it. If you are a user of WordStar, as I am, its failure to work with WordStar is obviously a serious issue, but not the only one. ATON CP/M is actually a year ahead of its time. The great selling point of ATON CP/M is that versions are available to support bank switching and high performance disk buffering. These will be standard features of CP/M version 3.0 which will begin showing up in 1983. Unfortunately, I have not tested these features because I don't have an extra board for my Model IIs, and the review copy I got was not the version that allowed use of the already-installed extra memory in the Model 16.

The thing that impresses me about ATON CP/M is its absolutely brilliant design. I can't really say they are original in their ideas, but their concept is a good one. What they have done is to take all the good features of Lifeboat CP/M and combine them with all the good features of Pickles & Trout CP/M, and add to that the features of CP/M 3.0 which isn't even on the market yet, and make the whole thing available for a very competitive price. Their disadvantage is that Lifeboat and Pickles & Trout have a three-year head start on bug catching.

An example of what I am talking about shows up in the otherwise nicely done installation and configuration programs. If, for example, you attempt to run the configuration program in the 24 line mode with clock on, you will find that the input line is forced off the bottom of the screen in some cases. It still works, but I found this behavior upsetting, even as an experienced user.

Another non-feature of the configuration program is its general refusal to accept lower case letters as input for menu selection where upper case letters were expected. Those of you who are TRSDOS users probably won't be offended by this, but as a dyed-in-the-wool CP/M user, I have difficulty excusing the existence of an alphanumeric menu that will reject a lower case letter. The extra programming effort required to make a menu selection routine oblivious to the upper-lower case distinction is too trivial to justify its omission.

Pickles & Trout

Pickles & Trout CP/M is regarded by many as the best of the CP/M variants on the market. I am one of those who is not so sure, however. My most serious criticism of P & T relates to the fact that it has no provision for emulating any industry standard terminal. This issue is perhaps becoming less significant as time goes by because its acceptance has been great enough to inspire many software vendors to simply include its terminal control conventions as another terminal to be accommodated. I can't help but think that standardization is a constructive goal, and the industry needed another set of terminal control codes like it needed another set of holes in the head. It is my opinion that the probability of getting off-the-shelf software to run under P & T CP/M is lower than it is with Lifeboat.

The Problem with Printers

I have practiced — and preached — the use of non-Shack printers for CP/M installations on the Model II/16, and the reason is simple. CP/M was designed on a teletype terminal and has built into its very essence a sequence to issue a linefeed character with each carriage return. Shack printers are built like typewriters and mechanically ratchet to the next line whenever the carriage is returned. The result tends to be double spacing. Nearly every printer on the market, except those from the Shack, have a switch to disable the auto-line feed feature. One of the challenges of writing a printer driver that will work on a Radio Shack printer involves filtering out the unwanted linefeed commands.

All three versions of CP/M attempt to provide support for the Shack printers. Over the years, both Lifeboat and P & T have had trouble getting the linefeed filter to work correctly. Since I have no Shack printers, my information is necessarily second-hand, but from the best I can gather, P & T has had the worst of the problems with the linefeed filter. I don't know if this bug is applicable to current versions or not, but, for example, the combination of Shack printers, P & T CP/M and Structured Systems software wouldn't print correctly in the past. (If you haven't heard of Structured Systems, they are a West Coast distributor of some rather good accounting software.)

The other issue that bothers me greatly about P & T CP/M relates to the disk compatibility in the double density format. I have no idea which sector size is better. In fact, both seem to work well. The point is that there is little excuse for P & T to be different from Lifeboat and ATON. It is, of course, manifestly unfair to accuse P & T of being the nonconformist as it had their 600K byte format on the market before either Lifeboat or ATON. All I can say is, as an owner of both Lifeboat and P & T CP/M, I find the disk sectoring incompatibility utterly exasperating. If P & T can write a utility that can read a TRSDOS diskette (and it has one), why can't it furnish something to read a Lifeboat or ATON diskette? Of course, Lifeboat is as guilty as P & T.

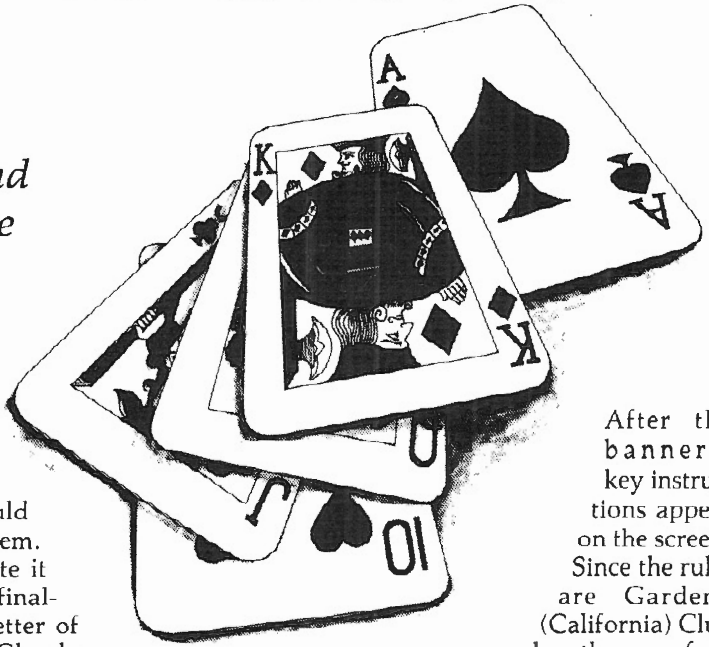
Conclusions

Lifeboat and P & T CP/M are both credible programs for use on the Model II/16. Both will make the Model II/16 into a CP/M computer that will, I believe, outperform many of the so-called "CP/M computers" on the market. Both have been on the market long enough so that you can reasonably expect their documented features to work. As with all programs, however, by the time one has been on the market long enough so that it works, it is also probably obsolete. The state-of-the-art CP/M version for the Model II/16 is obviously ATON. If you like surprises, it will give you some. Unfortunately, I am unable to assure you that they will all be pleasant.

See p.38 for more on ATON/WordStar.

DRAW5POKER

A Review of an Intriguing and Non-violent Computer Game



Robert Aaron, President
Aaron/Smith Associates, Inc.
P.O. Box 7567
Atlanta, GA 30357

I have trouble imagining people pumping quarters into the Model II in a video arcade. The frivolity associated with game-playing and the utilitarianism associated with the TRS-80 Model II seem to me to be mutually exclusive. Despite my love for the Model II's practicality, I have always bemoaned the lack of games for it. Most of the game advertised for other computers leave me cold (mythical monsters and eye-hand coordination games bore me very quickly), and I don't miss those kinds of games on my machine at all. But having virtually no games seems unreasonable; are Model II owners that dour a bunch?

Six Hands To One

That's why I was excited to see an ad for DRAW5, a Model II poker simulation from Wilson Software in Camp Hill, Pennsylvania. First of all, I am a big poker fan. Also, I was intrigued by the description of the game in the ad, which said that the computer played six hands while the user played one. I know from my own experience that it takes at least four players to make the odds interesting in a poker game; fewer than four and so few cards are used that no one can get a very good hand. In addition, I wanted to get a game to use while demonstrating my machine for my business clients. Finally, it was cheap (then $15, but now $20). So I ordered it. And I waited.

I waited almost three months. I tried to call Wilson Software but could find no listing for them. I was about to write it off when the game finally arrived with a letter of apology from Charles Wilson stating that he had been assured by his local Radio Shack Computer Center people that converting this program from the Model I to the Model II would be no problem, but that it hadn't worked out that way (I can sympathize). To compensate for the delay, Mr. Wilson also enclosed a copy of TOUT3, their horse handicapping program, which I appreciated.

After making a couple of backups of the program, I sat down to play; I think I played about three hours solid that first day.

Let me say at this point that I found DRAW5 to be worth the wait. Without hesitation I consider it to be a better-than-reasonably- good poker simulation, except for a few klunkers that I'll get to later. It has afforded me with many hours of pleasant diversion when I should have been doing something productive, and I mean that as a high compliment.

An Introduction to DRAW5

DRAW5 is completely in BASIC and takes about twelve seconds to load. Out of curiosity I listed the program but found it so tightly packed that I could hardly make heads or tails out of it. So I gave up trying to be a computer professional and typed RUN.

After the banner, key instructions appear on the screen. Since the rules are Gardena (California) Club rules, they are fairly standard: $1 ante, three raises before and after the draw, jacks or better to open (but it is not progressive if no one car open;e.g., jacks the first time, queens the second, etc. Also, if you can't open, you are automatically passed by), and so forth.

Five boxes outlined in reverse video with alphanumeric characters inside represent your five cards. Another identical box represents the pot, something I didn't realize immediately since nothing on the screen is labeled except the names of your opponents (Duke, Fats, Doll — you get the idea).

DRAW5 is a Winner!

DRAW5 shows evidence of having been well thought out. Consider the following features:

* DRAW5 will keep track of the amount of money in the pot, allowing for increments in case of raises, misdeals, etc., as well as keeping track of your current status of being ahead or behind X dollars.

* The deal is randomly assigned to a player when you run the program, anc then it progresses to the dealer's left. The card play goes to the left as well.

* Normally, the ante is $1 and the betting is $5 before the draw and $10 after.

II + II = (BUSINESS DYNAMICS)

DOSPLUS II and the TRS-80 model II. The Dynamic Duo.

The TRS-80 model II was created to help you take care of your business. DOSPLUS II was created to help your TRS-80 help you and your business even more.

DOSPLUS II replaces your O.E.M. dos with an ultra-sophisticated operating system that is 5 or more times faster, more reliable and is totally compatible with all standard business oriented software designed for use with the model II. The result is a computer/operating system that far exceeds manufacturer's specs—with more potential and more capacity to be of service to you.

DOSPLUS II operating systems are available for either your floppy or hard disk drives. The hard drive operating system is compatible with Radio Shack, QuCeS, Corvus Micro Mainframe, VR Data and other popular hard drive sub-systems.

DOSPLUS II... All the quality and dependability you'd expect from DOSPLUS.

For the dealer nearest you call
1-800-327-8724
Dealer inquiries invited.

$249.95

For product information (305) 983-3390

DOSPLUS DOSPLUS first in quality!
First in the industry!

To order call toll free 1-800-327-8724

POWERSOFT
A Division of Breeze/QSD, Inc.
11500 Stemmons Fwy., Dallas, Texas 75229
(214) 484-2976 Micronet 70130, 203

MSS inc.
MICRO SYSTEMS
SOFTWARE, INC.
4301-18 Oak Circle,
Boca Raton, FL 33431

If no one can open, everyone antes again, but now the bets are $6 and $12 respectively. This progression will continue until someone can open, and I've had it happen up to four times in a row — very realistic.

* If one player bets and another raises that bet, the appropriate total that comes to you is placed in the pot, if you bet. This is a feature that I wish real poker had; my friends and I always have to reconstruct our betting to figure out where we are. However, I would like to know how much that is *before* I bet.

* The draw is handled by DRAW5's asking you how many cards you want to discard. Once this is determined, you indicate which ones you want to trash, they disappear, and new cards appear to take their place once the discarding is completed. Incidentally, the cards are always displayed in ascending numeric order. DRAW5 will indicate how many cards are discarded by the other players still in the game, and then the betting resumes.

* Bluffing in betting by the computer's players is possible and is randomly assigned to the various players. Occasionally one player will bluff consistently, and other times several players will bluff once in a while. This feature keeps you guessing.

* If two players stay in until the bitter end, both hands are displayed and the program then determines and displays the winning hand. If all of the players drop out except one, no cards are displayed; you have to pay to see them.

* If you fold, one of the best features of DRAW5 is that the other players continue to play against each other. This really makes it seem like a real game.

* The algorithms used to make the evaluations of the hands seem to be reasonable. If a player (other than yourself, I should specify) stays in, most likely he/she/it has a respectable hand. The random bluffing feature doesn't seem to come into play too frequently. The feel of the game is surprisingly realistic, if you play as you would against human foes.

But if you don't act reasonably, well, the program doesn't either.

Nobody's Perfect

One of the touted features of DRAW5 is its supposed ability to "learn" from your actions and those of the other players; i.e., whether you and the others bluff repeatedly or infrequently. According to the ad and the description that comes with the game, DRAW5 has features that cause it to modify each player's style in response to the other players' styles. I have never seen any systematic evidence of this.

To test this, I tried raising on every bet for fifteen sequential games, regardless of my hand. No one ever caught on; everyone would drop out every time. If this program has such a feature, I can't imagine a clearer test case to trigger it, but it never changed. Apparently, you can always win through intimidation, and there's not much sport in that.

Other criticisms are in order here: A major complaint I have is that you can discard all of your cards at the draw if you like. This is not the way I was taught to play poker; I have always played that the most you could discard was three unless you had an Ace, in which case you could discard four. Never could you discard your entire hand; that was called folding. Also, if you specify that you will discard three cards, in this game you can specify the same card three times, in effect discarding only one card by doing it three times. One card comes back to fill out your hand. This is not allowed.

One of my irritations with the game is the way that it displays the cards. The following example is what a hand looks like (without the boxes around them):

```
4H 7D 10S KC AS
```

One gets used to this notation after a while, but I wish there were another way. It gets quite wearing on the eyes and brain, and I often miss potential flushes.

One eccentricity I don't especially like is the presence of a Joker that can be used as an Ace or as a wild card in straights or flushes. I consider wild cards to be intrusive little suckers whose presence serves only to mess up the strategy of the game.

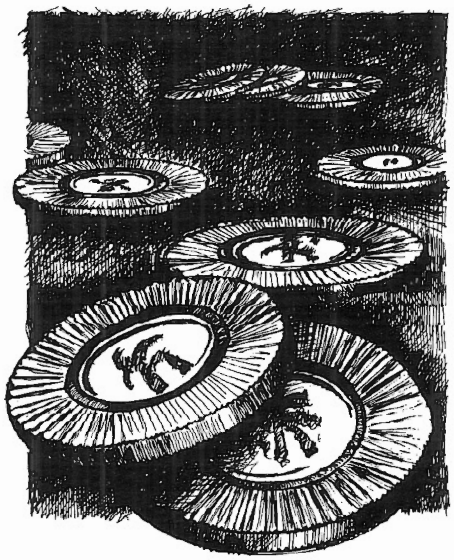
Another irritation I have is over the unreal real-time actions that happen regularly. Once the evaluations of the hands are made, the computer goes from player to player so quickly that all other six players can bid, raise, and drop in less than two seconds. This is unrealistic, disconcerting, and removes much of the tension that makes poker such an addicting game. This wait to see what others do is part of the real game but, unfortunately, is not a part of DRAW5.

Lastly, if I could rewrite the program, one feature I would add is a tally of everyone else's record in betting. Part of the fun of poker is the knowledge of where you and the other players stand relative to each other in winnings; it would be a simple matter to keep track of everyone's bets and consequences, not just the human's. As currently written DRAW5 doesn't do this, but maybe DRAW5 2.0 will (if there every is one).

While these drawbacks do occasionally detract from the overall excellence of the program, I have to say that DRAW5 is very satisfactory and a lot of fun as well. At $20, it's a steal and is easily worth twice that amount, especially given the dearth of available and intelligent games for this specific machine. I consider it tc be a prime example of how the power of the Model II can be used for the furtherance of leisure in the Western World.

Ante Up!

DRAW5 for the TRS-80 Model II:
Cost $20
Available by mail only from:
Wilson Software Division
539 Springhouse Lane
Camp Hill, PA 17011



REVIEW OF SNAPP-IV/SNAPP-V

Arnold Fischthal
Advanced Data Design, Inc.
184-08 Tudor Road
Jamaica Estates, NY 11432

In their quest to help the software developer produce better programs faster, SNAPP-WARE of Cincinnati, Ohio has a series of programming aids and utilities that make the life of the Model II and Model 16 programmer almost enjoyable. In this review I will be examining two of these products: SNAPP-IV, Extended Basic Mapping Support; and SNAPP-V, Extended File Mapping Support.

SNAPP-IV

As described on the cover of the product manual, SNAPP-IV is an "automated screen management facility for the BASIC interpreter." What does that mean?

SNAPP-IV routines allow the programmer to pre-define screen formats with up to 99 fields, and each specified field can have several attributes. When a programmer uses these attributes by specifying the maximum number of characters that will be allowed for a particular field and whether the field is alphanumeric or numeric only (if numeric whether negative numbers will be allowed and the maximum number of decimal places), he effectively reduces the possibility of error that exists when an operator enters any required information.

Each entry on the screen has a caption (e.g., last name) that can be pre-specified to be in normal or inverse video (as can the data portion of the entry). In addition, the data can be operator protected to display on the CRT only without the user being able to change it. The variable must be typed ($,%,!,#) and may be any legal variable, such as A$, AB#(9), CD%(I+J). The variable name may not exceed 11 characters.

The screens themselves are fairly easy to generate. Several editing commands allowing insertion, deletion, editing, and so on, of the fields are conveniently provided. The arrow keys move the fields around on the screen until you are pleased with the way the screen looks.

Once generated, the screen is saved in a disk file, and its use requires very little program space. The screen is accessed and displayed on the CRT from within the program by:

```
SCMD "INIT", screenname
```

The data is sent to the screen by

```
SCMD "SEND"
```

Data is entered onto the screen by the operator by:

```
SCMD "RECEIVE"
```

Unfortunately, a few minor drawbacks are included with all this ease of development. The time that expires from the point at which the screen generating program is run, to the time the screen has been stored, seems excessive. This is particularly apparent when you want to go back and edit a previously created screen. If an error occurs that forces you to change something in a particular screen, you must save the program under development, re-run the screen generating program, make the necessary changes, and then re-save the new screen. This process might take 2-5 minutes just for some simple cosmetic or reparative surgery — even the process of changing only one byte.

I have rarely run into any problems, but when I did, they were at times difficult to track down. One of these hard-to-find problems occurred when I inadvertently typed in a variable name that was too long. The screen generating program did not advise me of this error, and only the first 11 characters of the variable name were saved. When the program was executing, I got a syntax error in Line 1000. Line 1000 was simply SCMD "SEND" which evidently was typed in correctly. It took awhile for me to realize that the screen itself had the error, and I had to go back into the screen to make the corrections.

In summary, the main benefit of SNAPP-IV (which I believe far outweighs the drawbacks mentioned above) is the reduced coding: coding necessary for screen formatting and coding required to check for incorrect operator entries. Not only does this reduction of required coding save time, but it can greatly reduce the size of the completed program.

The documentation and performance of this product rates a 10; however, its error trapping (that is, trapping the programmer's errors) and ease of use rate a 9.

SNAPP-V

As described in the product manual, SNAPP-V is an "automated disk I/O management facility for the BASIC interpreter." The routines in SNAPP-V replace BASIC's FIELD and LSET with commands that are much more flexible. The best way to describe them would be by example; first, here's a BASIC program:

```
10 OPEN "R",1,"DATA",90
20 FIELD #1, 10 AS A$, 20 AS B$, 10 AS C$(1),
     10 AS C$(2), 10 AS C$(3), 2 AS D$, 4 AS E$,
     8 AS F$, 8 AS G$(1), 8 AS G$(2)
30 GET #1,1
40 AA$ = A$:BB$ = B$:CC$(1) = C$(1):CC$(2) = C$(2):CC$(3) =
     C$(3):D% = CVI(D$):E = CVS(E$):G#(1) = CVD(G#(1)):
     G#(2) = CVD(G$(2))
```

When PUTing data, you would need all the corresponding LSETs.

The new code for the above program would be:

```
10 OPEN"R",1,"DATA", 90
20 FD$="(10)A$,(20)B$,(10)C$(1);C$(3),D%,E,G#(1);
   G#(2)"
30 SCMD GET FD$,1,1
```

To PUT the data, all you need to type is:

```
SCMD PUT FD$,1,1.
```

I discovered two extremely beneficial results from using this code:

1.) As evidenced by the above example, there is a good minimization of the code necessary to perform normal disk I/O. This code reduction makes smaller programs, and

2.) No longer is it necessary for the code to force the variable in and out of file buffers. This function, plus the formatting of numerics (%,!,#), is now handled automatically — no more CVI, MKS$, etc.

OTHER GOOD POINTS

Less fragmentation of string space occurs since the need to take the string out of the buffer area no longer exists.

If a particular disk file contains different kinds of data (that would normally require reFIELDing the file and a new batch of LSETs), then more than one file descriptor (the FD$ variable above is the file descriptor) can be used. For example, if record number 1 contains name and address data, we would

```
SCMD GET FD$(1),1,1
```

and if record 2 contains accounting data, we would then

```
SCMD GET FD$(2),1,2
```

We would use PUT twice to store the data. (Note to users and potential users: if you do perform successive GETs, and you want to retain the strings in the buffer from the first GET, you must move those values out into string space.)

A new data type, the one byte integer, is now available for use in the field descriptor and can be used to store integers in the range of 0-255 only. It is specified in the descriptor by e.g., (1)IN%. This eliminates the necessity to:

```
IN$=CHR$(IN%)
```

and then

```
IN%=ASC(IN$)
```

since this is now handled automatically.

One final benefit of SNAPP-V is that the Radio Shack Renumber command has been replaced with SNAPP's ENHANCED RENUMBERING FACILITY (XRENUM is also found in their SNAPP-II). The space normally occupied by the Tandy Renumber in the BASIC interpreter has been used to optimize the performance of SNAPP-V commands.

Among its other properties, I have found the following to be most useful: SNAPP-V allows the programmer to specify an upper limit to a block of code to be renumbered. It allows blocks of lines to be relocated in the program. It provides the capability to duplicate portions of the program elsewhere in the program.

XRENUM also corrects some deficiencies of Radio Shack's Renumber. Without renumbering, it allows you to search the program for non-existent lines that are referenced. If you want, it can renumber without altering any references to such lines. This might be necessary if you have referenced routines which have yet to be added to the text.

Moreover, whenever XRENUM is used, you get two informational tidbits: how many lines in the program and how many bytes the program occupies. The module is accessed by typing:

```
R newline, increment, startline, endline
```

In summary, as with SNAPP-IV, SNAPP-V also reduces coding and coding time. Unlike SNAPP-IV, there are no drawbacks. All four of the tested areas — (1) documentation, (2) performance, (3) error trapping, and (4) ease of use — rate a 10.

***

As a final note, I would like to say that the customer support and service offered by Bob Snapp and his company are excellent. This fact should weigh heavily in the possible purchase/use of any software package: does the company stand behind its product, not simply with a money-back guarantee, but with the time and energy it takes to properly service a client? With SNAPP-WARE, there is never a question; the answer is always "yes."

SNAPP-WARE PRICES:

SNAPP IV	
MODEL II HARD DISK	$150.00
MODEL II FLOPPY DISK	$100.00
SNAPP V	
MODEL II HARD DISK	$115.00
MODEL II FLOPPY DISK	$ 75.00

THE MEDICAL MICRO

Robert Dale Langer, M.D.
The Center for Health
16766 Bernardo Center Drive,
Suite 111-A
San Diego, CA 92128

This is the first edition of what we hope will be a regular feature in this publication. Loving gadgets as we do, no small number of physicians have purchased microcomputers in the past few years. If my experience and that of the many physicians with whom I have discussed these little gems is at all indicative, microcomputers have been a source of both tremendous excitement and frequent frustration amongst members of the medical profession.

A large number of doctors seem to have purchased micros with a specific purpose, such as patient billing, appointment scheduling, or medical record keeping, in mind — only to find that the salesman's or programmer's concept of that function in a medical office was so far from reality that the machine quickly became a white elephant or was delegated some other less exciting function for which general purpose software was readily available, such as word processing or financial modeling.

The TRS-80 Model II was my first micro; I bought it with bright eyes and a whole slew of applications in mind for my solo family practice in San Diego. like most of you, I was impressed by its capabilities and somewhat naively accepted the salesman's assurances that the software available would easily and quickly be adapted to perform the functions I had envisioned. Right!!!

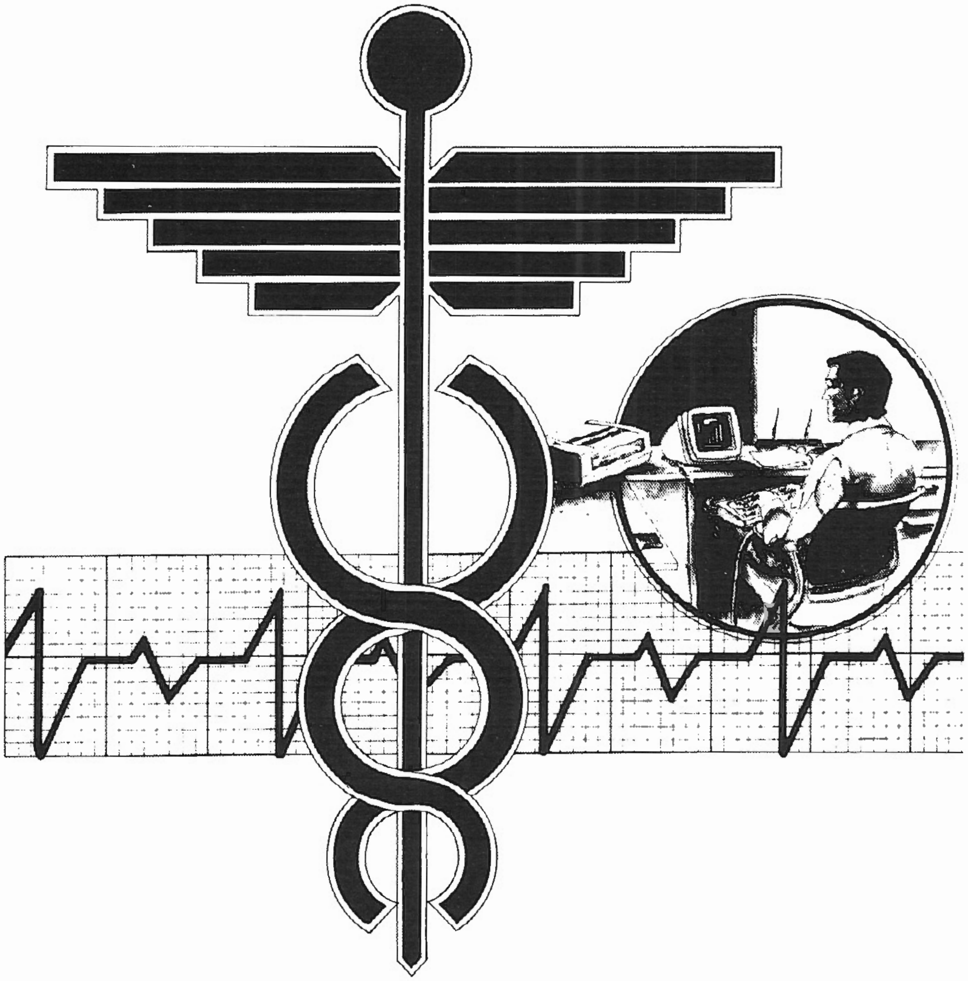
It is now nearly two years later, and I have long since lost that early naivete. One of those early assumptions continues to hold true, however; the Model II is a marvelous and powerful machine. And the Model 16, if and when it ever has an operating system that will allow it to do what it has been designed to do, is easily one of the most powerful and useful micros yet produced.

The Model 16 fiasco is just the latest of many instances in which we Model II and 16 owners have been orphaned by Tandy and the microcomputer industry in general. I hope in this column to discuss available and potential serious applications for medical office management and other aspects of patient care which have been developed for, or can be adapted to be used on, the Models II and 16. I intend to address both hardware and software issues and to discuss newer developments in the incredibly fast moving field of microprocessors which may prove useful in medical environments.

Your comments and feedback will be invaluable to me — let me know what you have done with your machine, what you would like to be able to do, what software or hardware you have used, and how it has or has not accomplished your goals. We will all benefit by pooling such experiences; perhaps some of us will be spared the misfortune of purchasing a product unsuited to the purpose we had in mind.

* * * *

My initial interest in microcomputers for my practice involved medical record keeping applications. Microcomputers have found their way into physician's offices largely as accounting machines. While there is certainly nothing wrong with improving cash flow, it seems a terrible waste of such a versatile resource to stop there. If anyone else could read our handwriting we physicians would be terribly embarrassed to admit that we write our daily patient care notes in a format designed to be used as part of a data base — but we don't have the data base to plug the information into!



Aw, what the heck, let's let everyone in on the secret — for more than a decade virtually every medical school in the country has taught its students to organize their patient care notes in what is know as the Problem Oriented Medical Record (POMR). What is the POMR? It's a modified data base with two major segments; a Problem List and a Medication List. All narrative regarding a patient is theoretically assigned to a Problem and recorded in a four part logical sequence — Subjective (history), Objective (physical or laboratory findings), Assessment and Plan. This format is affectionately known as SOAP (for the first letter of each major section). It was originally developed to be used with computers but was never implemented on any major scale because the mainframes and even minis which were required to run the software were generally not available to the physicians who took care of patients! We now have a whole generation of physicians who have learned their profession using an advanced conceptual framework which virtually none have been able to take to its logical (and original) extent. Improved collections may be an exciting use for computers in medicine, but what if we add improved patient care and the ability to extract an almost unlimited number of relationships from progress notes that presently do little other than collect dust? One could select patients requiring flu shots, pap smears, and other routine interval examinations, find all patients taking a drug for which new warnings have been issued, monitor the demographic characteristics and common problems in one's patient base, select topics and extract data for research, even do some epidemiology ... the possibilities are almost limitless! And what if we could accomplish all this without having to learn new ways of conceptualizing or recording data and ... Oh gosh! I just realized why there hasn't been a flood of these programs; other people would be able to read our notes! My heavens!

All kidding aside, the power and cost-effectiveness of microprocessors now provides the opportunity to implement the POMR in daily practice. The software to perform these functions requires a hybrid approach which includes elements of both data base management and word processing. Considerable archive capacity is also necessary to store the text of progress notes.

A number of practical solutions are possible. One approach which some physicians have adopted is to use a word processor and write their notes in a problem-oriented format. The disadvantage of this approach is that data base functions in word processing software are quite limited — if available at all. The text of the records entered is still practically inaccessible for analysis on a macro scale. Standard TRSDOS and CP/M compatible data base software is also unsatisfactory because text for progress notes cannot be stored efficiently, the storage available in "key fields" (those which can be looked at in a search) is inadequate to the complexity of medical records, and because each patient may have multiple problems which must each be weighted equally in a search. For example, hypertension may be the first listing for one patient and the fifteenth for another, but must be found and evaluated as equivalent in both of these records when they are scanned for patients with this problem. While it is true that some data bases may be able to find these relationships eventually, they are so cumbersome and inefficient in doing so that it is unlikely anyone would ever spend the time to perform such a search. The logistics become virtually insurmountable when a search for multiple parameters is desired.

Having experimented with a number of word processors and data bases soon after purchasing my Model II, I was rapidly forced to the conclusion that if I wished to implement a truly useful Problem Oriented Medical Records system, I would have to write the software myself. I originally developed a version to run on my Model II together with an integrated office management system. I'm sure my experience is akin to the experience of others who naively began to develop their own applications, my original estimate of six weeks to have the system up and running stretched into six months. Our first "rough draft" working system was "hard-coded" to contain the kinds of information I routinely used in my practice (in other words the phrases, problems, medications, diagnoses and so on were all contained in the program itself rather than a reference file). This method proved impractical from two stand points: first, the diagnoses or medications which were not contained in the program could not be used; and second, as my colleagues saw the system and became interested in using it themselves, they were forced to either rewrite a major portion of the program code or use my personalized system.

We have since developed enhanced versions which allow any physician in any specialty to configure the system to his personal style of practice. Moreover, the entry routines are set-up to allow the system to "learn" new problems, medications, billing codes, and diagnoses, while working on an individual patient's record so that it remains practical to use in "real time" in a busy office.

The lack of multi-user capability on the Model II proved a serious drawback, however. Inevitably my receptionist would be running the billing routines when I wanted to work on medical records or vice versa. This led me to purchase a second micro, an LNW-80 (a Model I/III compatible machine with several improvements), to which I downloaded the functions I used most often, including the medical records routines. Having a computer which could be dedicated to these functions allowed the development of enhanced versions of the medical records programs which allow the archiving of the full text of progress notes for every patient visit ad infinitum, as well as the review of such notes by problem and in chronological order. These enhanced POMR systems run under the Microsoft BASIC Interpreter on the Model I, Model II, integrated medical office management system requires the capacity of the Model II.

More on some of the uses of these systems as well as many other things in later columns. Till next time...

CROSS REFERENCE LISTING

What Is It?...How Is It Used?

Joe Lyddon
Good-Lyddon Data Systems
5486 Riverside Drive
Chino, CA 91710

What exactly is a cross reference listing? If you are like many people involved in programming, you don't know what it is, or how to use it.

Simply stated, a cross reference listing is a valuable and extremely useful tool for designing, writing, debugging, maintaining, and changing programs. It's a printed report that locates, counts, organizes, and then lists the data found in your program. A cross reference listing tells you the variable names used and where they were used, the transfer points used and from where they received transfers, and the reserved words used and where they were used. In certain circumstances, a cross reference listing will show the numeric literals used and where they were used. A good cross reference listing produces a program print-out that has the date and time in the page headings. The programming language used doesn't make any difference; the cross reference listing works with COBOL, FORTRAN, ASSEMBLY, BASIC, or XYZ language.

In this article, we'll use the RS BASIC interpreter language cross reference as an example. In BASIC, line numbers are used as transfer points: GOTO...,IF A 0 THEN...ELSE .., etc. When you want control to branch to a statement other than the next one in line, you reference a line number. In BASIC, the line number is the main point of reference.

Let's analyze the following small program:

```
100 A$="":B$="":C$=""
110 INPUT A$
120 IF LEN (A$)=0 GOTO 150
130 PRINT A$
140 GOTO 110
150 END
160 LPRINT A$
170 RETURN
```

A full cross reference report of this program would be:

```
Line #s    Referenced at Line #s

  110                     140

  150                     120

Names:

A$        100 110 120 130 160

B$        100

C$        100

END       150

GOTO      120 140

IF        120

INPUT     110

LEN       120

LPRINT    160

PRINT     130

RETURN    170
```

The first thing you will see on the cross reference listing is that only line numbers 110 and 150 have been referenced. Thus you know immediately that these lines cannot be deleted without changing other lines that reference them. What a useful piece of data to have! It can save you hours of tiresome searching. If you want to delete a section of code, you can easily modify and discover other code that uses it. On the other hand, you may choose not to delete.

A hard-copy print-out of the various names used in your program can be helpful. In our example, there are only three variable names; the rest are reserved words. If a variable name is referenced only once, it may be useless and should probably be removed. For example: B$ and C$ are only referenced in line 100. They are dead names and should be removed, or your program should be analyzed further. Perhaps you meant to use them and forgot.

Another handy use for your cross reference listing is to search for dead code (code that will never be executed).

Examples of DEAD-ENDS are:

```
1000 GOTO.......
 . . . .
 . . . .
2000  RETURN
 . . . .
 . . . .
3000 END
 . . . .
 . . . .
4000 IF XXXXX THEN .......ELSE......
 . . . .
 . . . .
```

The line number immediately following a dead-end must be referenced to be of any value. All unreferenced lines following a dead-end are definitely dead and can be safely deleted. Before deleting, however, you should take into consideration the possibility that you forgot to use the code. Check it out.

In our example, lines 160-170 are dead. What if you had intended to incorporate an optional printed report of A$ data and forgot to input the code? If you still want to incorporate an optional printed report of A$ data, you can insert the necessary code, or you can delete. Because you used the cross reference report, you were able to isolate the problem and take action.

As a programmer, you are probably on the look-out at all times for ways to improve your programs and make them more efficient. The cross reference report can help. Start by circling each reference line number on your program listing. (In the example 110 and 150 are the reference line numbers.) All code from one referenced line up to the next referenced line may be changed or deleted without affecting present line references.

In our example, the following code can be changed:

```
110  INPUT A$
120 IF LEN(A$)=0 GOTO 150
130 PRINT A$
140 GOTO 110
```

Since you know that no other program changes are required, the following change could be made:

```
110 INPUT A$:IF LEN(A$)()0
  THEN PRINT A$:GOTO 110
```

Lines 120 through 140 have been safely deleted.

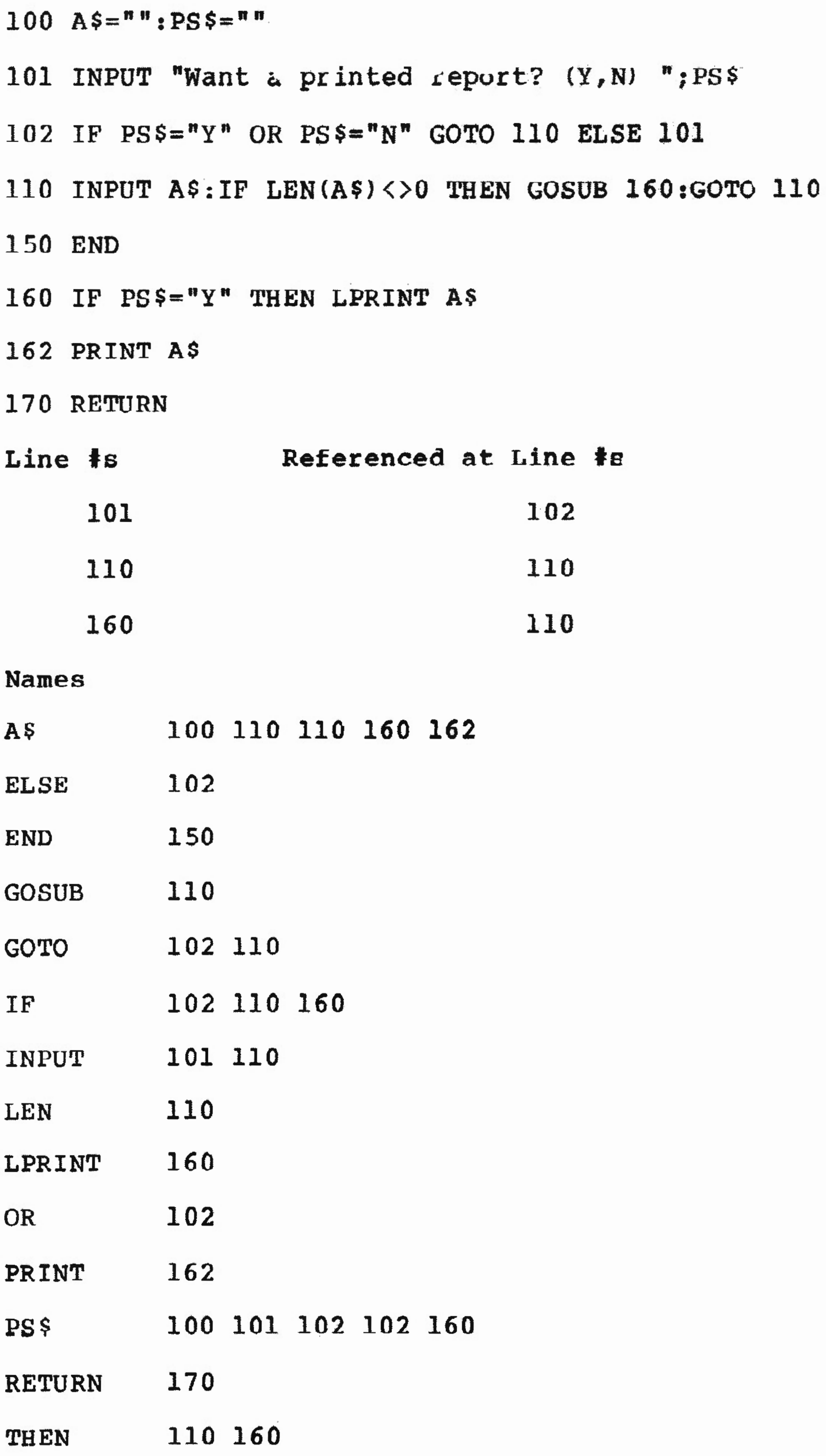
A cross reference report can be very helpful when you want to add code to your program. For example: You want to add the option of printing a report, and you need to use a new variable name that can be used for a print switch. PS$ would be the logical choice, but has it been used before in the program? A quick glance at the cross reference report answers the question. No, the variable name PS$ has not been used previously so go ahead and use it.

In this example, determining the status of PS$ was easy. In more complex programs, however, the task of finding out whether certain code names have been used before can pose a serious problem! You don't want to inadvertently use a name that is already being used!

A word of caution: RS BASIC recognizes only the first two characters of a name. For example: AA1$, AA2$, and AA$ will all be processed as AA$. You should restrict your names to no more than two characters. If more than two are used, a quick scan of the cross reference report will reveal any conflicts.

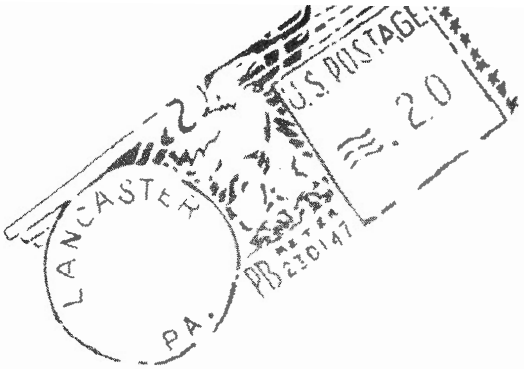
The newly modified program and its cross reference listing now appears as shown on this page.

You may question the inclusion of reserved words in the cross reference report. I believe they should be included because the cross reference report becomes a more efficient debugging tool when they are included. OPEN, CLOSE, FIELD, DEF, DIM, LPRINT, INPUT, GET, PUT, PRINT@, STOP, END, RUN, LSET, RSET, LOF, KILL, and ERASE can be very helpful when debugging a complex program. On the other hand, IF, THEN, ELSE, GOTO, GOSUB, MID$, LEFT$, RIGHT$, etc. probably won't be very useful to you in your debugging efforts. But you can easily ignore them if you want to, and someday you may be very thankful they're on the cross reference list!



```
100 A$="":PS$=""

101 INPUT "Want a printed report? (Y,N) ";PS$

102 IF PS$="Y" OR PS$="N" GOTO 110 ELSE 101

110 INPUT A$:IF LEN(A$)<>0 THEN GOSUB 160:GOTO 110

150 END

160 IF PS$="Y" THEN LPRINT A$

162 PRINT A$

170 RETURN

Line #s                Referenced at Line #s

     101                              102

     110                              110

     160                              110

Names

A$         100 110 110 160 162

ELSE       102

END        150

GOSUB      110

GOTO       102 110

IF         102 110 160

INPUT      101 110

LEN        110

LPRINT     160

OR         102

PRINT      162

PS$        100 101 102 102 160

RETURN     170

THEN       110 160
```

Feedback



From: H. Lawrence King, 365 West End Avenue, New York, NY 10024:

1.) If portions of *two/sixteen* are still printed on a dark gray background when renewal time comes around, count me out. I won't buy what I can't read.

2.) Please define all abbreviations. "Computereze" is hard enough to comprehend even when it is spelled out.

3.) Give some indication of the cost of the items you review; i.e., SNAPP-III sounds great at $50 but not at $500.

4.) I have a subscriptions to *Data Cast* (recommended by you for their tutorials). They appear to have ceased publication.

5.) Please keep expanding and improving. You are very important in an industry where few suppliers provide enough information to use their products.

Ed.:
1) This was a mix-up at the printers. Many of our screens were printed much darker than we indicated. As a result of this (and unsatisfactory printing schedules, as well) we are probably changing printers with this issue.

(2) We try to do this, but its easier said than done. It's difficult to gauge which abbreviations are known by most readers and which aren't. We can reasonably assume that all readers know the meaning of TRSDOS or CP/M, but should we define BIOS or SVC every time it first appears in a given article? We do watch for terms which are likely to be arcane to the readers, but we'll probably continue to miss some of them.

(3) It is our intention to include prices with all reviews. This one was overlooked. The price for SNAPP-III is $100.

(4) We understand that **DataCast's** *fifth issue (the first one since March/April) is at the mailers and should be mailed by the end of the week (January 1).!*

From: R. D. Schalow, Analytic Disciplines, Inc., 2410 Coliseum Blvd. North, Suite 108, Ft. Wayne, IN 46805:

I have just reviewed my first issues of *two/sixteen* and they are terrific! Here's hoping and wishing that your magazine is an unqualified success and that you make piles of money — well earned, I might add. I had intended to let some of my associates review my own copies, but because there is so much useful and to-the-point information, please find two more subscriptions enclosed.

Some random comments:

-Rick Rump (MicroShell) worked for us until recently. A fine, sharp person, Rick wanted to develop such products under the Company banner. Because this type of thing is not our line (maybe it should be) I encouraged Rick to leave and form his own operation. While we hated to lose him, we're all delighted to watch his progress, and it's good to see reviews such as yours which praise his efforts.

-The official support to be given by Tandy to independent software developers (*TRS-80 Microcomputer News*, December 1982) is very welcome news. Better late than never. But I hope this doesn't portend more delays in supplying Tandy-developed software.

-Please consider publishing an annual index to articles in the magazine.

-If for any (unimaginable) reason you get to the point where you feel breakeven status will not be achieved in a timely manner, please advise your subscribers *before* pulling the plug. I'm certain they won't willingly let such an event occur.

Ed.:
We will consider an index, but probably not publish one until the end of 1983. Since there were only three issues in 1982, a separate index for that year doesn't seem warranted.

We are still growing at a slower rate than hoped for, but with continued use of direct mail promotion, we may reach break-even by the end of 1983. However, our low subscriber volume (2224 at late December) does not encourage advertisers, and our revenue won't support an adequate number of personnel (or printing costs) for further expansion of the magazine. We could send out renewal notices six months early like **Creative Computing** does, but this is repugnant to me.

A new magazine, ***MainStream Microcomputing***, *and a newsletter (tentatively titled* ***Microcomputer Early Warning Letter****) for senior corporate management, are planned for introduction in the near future. More about them is included elsewhere in this issue. We encourage our readers to consider subscribing to these magazines!*

From: Norman N. Holland, James H. McNulty Professor of English, 409 Samuel Clemens Hall, State University of New York at Buffalo, Amherst, NY 14260 (9/1/82):

. . .I think the magazine is off to a fine start. I was tremendously impressed with the DeBerry-LaPorte piece on programming the video, a possibility hitherto closed to me.

. . . please don't forget those of us who are still locked into TRSDOS 1.2. I've got 50 disks on Electric Pencil and TRSDOS 1.2 and there just isn't any worthwhile way to change them and no reason to — everything works fine as is. Isn't that an old computer maxim? If it isn't broke, don't fix it?

Ed.:
That's an old computer maxim all right, but its interpretation depends on your definition of "broke." Here is another old computer maxim:

Always, always convert to the latest available operating system as soon as possible. The penalty for unbelievers is the eventual requirement to do a multiple-step conversion later, or to be unable to run any new software (including language enhancements).

From: William P. Herbert, Atlantic British Parts, Ltd., Box 109, Mechanicville, NY 12118 (7/27/82):

Thanks for the new publication. It is sorely needed.

I would really like to see the following information:

1.) What is going on with ARCNET?

2.) When, why and how of software for multiple users, etc. on the Model 16.

3.) Hard Disks for Model II/16.

Surely by this time it should be obvious that a large, fast mass- storage device is necessary for many interactive uses. Murphy's Law dictates back-up and floppies is *NOT* the way to do it, nor is fixed removable ("Hum, let's see. Was that 1 to 0 or 0 to 1? Whoops!"). With so many manufacturers now offering data cassette back-up integral with a disk, it seems that this is the way to go.

Ed.: (1) According to Radio Shack sources ARCNET is still "on hold" as of January. There is no indication as to when it will proceed. However, if you need a network in the near future, you might investigate the Corvus OmniNet. It has been in commercial use for several years and is produced by Corvus Systems, 2029 O'Toole Avenue, San Jose, CA 95131. Their telephone is (408) 946-7700.

(2) Radio Shack is expected to release Microsoft's XENIX version of UNIX in February. In addition, it has been rumored that a Ryan-McFarland multi-user operating system will be available from Radio Shack in the near future (in fact, some attendees at the Las Vegas Comdex indicated that RS was running this system at their booth). Other vendors are likely to announce operating systems for the 16 within the next two or three months. A CP/M 68 (?), a p-code, and other UNIX versions are frequently mentioned, but vendors are not committing themselves.

(3) Hard disks, of course, must be supported by an operating system. At present small hard disks (8 to 20 megabyte) by Radio Shack, Corvus, Cameo, and others are supported by various operating systems, including TRSDOS, Pickles & Trout CP/M, and Aton CP/M. However, all of these presently support only the Z-80 side of the Model 16. Use of the 68000 chip with hard disk requires a new operating system. In addition, there are indications that substantially larger (100 to 140 meg) hard disks will be available in the near future at prices about the same as present small models. These new hard disks will require some new form of backup facility. (Can you imagine trying to backup to 100 floppy disks?) A previously reliable source foresees a new RS 40 meg hard disk by summer, but has nothing about price or backup facilities.

SNAPP-WARE

3719 Mantell Ave.
Cincinnati, Ohio 45236
(513) 891-4496

ichard H. Young
ditor
wo/sixteen
31 East Orange Street
ancaster, PA 176Ø2

e: July-August review of PRO-8Ø BASIC.

ear Dick:

 am writing to direct your attention to a misprint in the sort timings in his review. When I first read the chart, my reaction was "HA! A little low? I can sort faster than THAT in BASIC!" and I rushed over to clean the osmoline from a good one I used to use with compiled programs. Sure nough, It was able to beat every time handily.

 was getting ready to write you a letter which contained a big put-down of ROBASIC when the realization hit me that the timings just couldn't be correct. Even a BUBBLE sort (faster only than a turtle) would beat the timings ou presented. I then proceeded to resurrect a copy of PROBASIC which the author had kindly sent me some time ago. Sure enough, the timings were wrong. I believe that the numbers at the top of the chart should have been 1ØØØ, ØØØ, and 4ØØØ rather than 1ØØ, 2ØØ, 4ØØ. As your reviewer notes, it is not s fast as some, but it is not as shabby as your error made it appear to be.

 think you owe PRO-8Ø Systems a retraction/correction/apology.

erhaps your readers might be interested in the relative performance of a eally speedy sort (ours). These are the average of ten runs, using RSDOS-II v4.1:

char	1ØØØ	2ØØØ	4ØØØ
2	2.9	5.9	11.2
3	3.7	7.5	16.Ø
4	3.7	7.7	17.2
5	3.8		
6	3.6		
8	3.7		

Your readers might be interested to know that while the SHELL algorithm is certainly much quicker than linear exchange (e.g. BUBBLE) type algorithms, it is the SLOWEST of the generally used non-linear algorithms. My favorite book on the subject is SORTING AND SORT SYSTEMS, by Harold Lorin (Addison-Wesley, 1975).

Since I had the old BASIC sort dusted off, I will send it along for your readers entertainment. I don't recommend sorting in BASIC unless you are forced to (like in a compiled program), but the algorithm contained herein is the fastest one I have ever seen. This algorithm was presented to the ACM by Richard C. Singleton in 1968, and was an extension of work done by Hoare and Scowen.

First, the driver program, which simply builds a random array and invokes the sort. The process is repeated ten times, printing the start and end times of each sort invocation. Set the variable AS% to select the size of the array, and the variable SS% to set the length of the strings.

```
1ØØØ CLEAR 2ØØØØ
1Ø1Ø DEFINT A-Z
1Ø2Ø FOR K% = 1 TO 1Ø
1Ø3Ø   RANDOM
1Ø4Ø   AS% = 1ØØ
1Ø5Ø   DIM A$(AS%)
1Ø6Ø   SS% = 2
1Ø7Ø   WS$ = STRING$(SS%," ")
1Ø8Ø   TP%(Ø) = 1
1Ø9Ø   BO%(Ø) = AS%
11ØØ   FOR I% = 1 TO AS%
111Ø     A$(I%) = WS$
112Ø     FOR J% = 1 TO SS%
113Ø       MID$(A$(I%),J%,1) = CHR$(47 + RND(1Ø))
114Ø     NEXT
115Ø   NEXT
116Ø   Q% = FRE(A$)
117Ø   PRINT TIME$ ;" ";
118Ø   GOSUB 1ØØØØ
119Ø   PRINT TIME$
12ØØ   ERASE A$
121Ø NEXT
122Ø END
```

Now the sort itself:

```
10000 ST% = 0
10100 LO% = TP%(ST%)
10200 HI% = BO%(ST%)
10300 IF (HI% - LO% > 13) THEN
        ELSE 12800
10400 ME% = (LO% + HI%) \ 2
10500 TI% = LO%
10600 BI% = HI%
10700 IF A$(LO%) > A$(ME%) THEN SWAP A$(ME%),A$(LO%)
10800 IF A$(HI%) < A$(ME%) THEN
        ELSE 11100
10900 SWAP A$(HI%),A$(ME%)
11000 IF A$(LO%) > A$(ME%) THEN SWAP A$(LO%),A$(ME%)
11100 PI$ = A$(ME%)
11200 BI% = BI% - 1
11300 IF A$(BI%) < = PI$ THEN
        ELSE 11200
11400 TI% = TI% + 1
11500 IF A$(TI%) > = PI$ THEN
        ELSE 11400
11600 IF TI% < = BI% THEN SWAP A$(BI%),A$(TI%)
11700 IF TI% > BI% THEN
        ELSE 11200
11800 IF (BI% - LO%) > (HI% - TI%) THEN
        ELSE 12300
11900 TP%(ST%) = LO%
12000 BO%(ST%) = BI%
12100 LO% = TI%
12200 GOTO 12600
12300 TP%(ST%) = TI%
12400 BO%(ST%) = HI%
12500 HI% = BI%
12600 ST% = ST% + 1
12700 GOTO 10300
12800 IF ((LO% + 1) - (HI%)) > 0 THEN 13700
        ELSE FOR I% = LO% + 1 TO HI%
12900 PI$ = A$(I%)
13000 TI% = I% - 1
13100 IF A$(TI%) > PI$ THEN
        ELSE 13600
13200 A$(TI% + 1) = A$(TI%)
13300 TI% = TI% - 1
13400 IF A$(TI%) < = PI$ THEN
        ELSE 13200
13500 A$(TI% + 1) = PI$
13600 NEXT
13700 ST% = ST% - 1
13800 IF ST% < 0 THEN
        ELSE 10100
13900 RETURN
```

The sort code show above is uncompressed, for readability by human beings. The computer, of course, prefers it compressed, like this:

```
10000 ST%=0
10010 LO%=TP%(ST%):HI%=BO%(ST%)
10020 IF(HI%-LO%>13)THENELSE10150
10030 ME%=(LO%+HI%)\2:TI%=LO%:BI%=HI%
        :IFA$(LO%)>A$(ME%)THENSWAPA$(ME%),A$(LO%)
10040 IFA$(HI%)<A$(ME%)THENELSE10060
10050 SWAPA$(HI%),A$(ME%):IFA$(LO%)>A$(ME%)THENSWAPA$(LO%),A$(ME%)
10060 PI$=A$(ME%)
10070 BI%=BI%-1:IFA$(BI%)<=PI$THENELSE10070
10080 TI%=TI%+1:IFA$(TI%)>=PI$THENELSE10080
10090 IFTI%<=BI%THENSWAPA$(BI%),A$(TI%)
10100 IFTI%>BI%THENELSE10070
10110 IF(BI%-LO%)>(HI%-TI%)THENELSE10130
10120 TP%(ST%)=LO%:BO%(ST%)=BI%:LO%=TI%:GOTO10140
10130 TP%(ST%)=TI%:BO%(ST%)=HI%:HI%=BI%
10140 ST%=ST%+1:GOTO10020
10150 IF((LO%+1)-(HI%))>0THEN10200ELSEFORI%=LO%+1TOHI%
10160 PI$=A$(I%):TI%=I%-1:IFA$(TI%)>PI$THENELSE10190
10170 A$(TI%+1)=A$(TI%):TI%=TI%-1:IFA$(TI%)<=PI$THENELSE10170
10180 A$(TI%+1)=PI$
10190 NEXT
10200 ST%=ST%-1:IFST%<0THENELSE10010
10210 RETURN
```

Naturally, I am delighted to have a magazine taking my favorite machines so seriously. Keep up the good work.

Kindest Regards,
SNAPP, INC.

Bob

Robert F. Snapp
President

WHAT'S AN SVC??

Jim Kloosterman
Post Office Drawer 48
Rogers City, Michigan 49779

Before I tell you what an SVC is, let me tell you what it is not. No, an SVC is NOT yet another toxic chemical, government agency, or some electrical heart malfunction. Look near the end of the TRSDOS Section of your Owner's Manual, in the part entitled "Technical Information". Note the many pages devoted to named routines with function codes. THOSE are SVCs — Supervisor Calls. If you'll look in the "Technical Information Section" of your Model II Owner's Manual, you'll see a page or two on "How to Use the Supervisor Calls."Don't find it very illuminating? Don't give up; page through the routines again. Aren't you intrigued by some of the "sneaky" things they can do?

Why do the SVCs exist? Why are they in your Owner's Manual? All Radio-Shack (R/S) software, including TRSDOS, has occasion to do many little things over and over again. It is a great time-saver in Radio Shack's program development to have a standard set of routines for those repetitive chores. The programmers at R/S may use the SVCs rather than "reinvent the wheel" every time they have a programming job to do. That's why SVCs are in RAM! Although they were not originally intended for your use, R/S has unveiled some of them.

Who uses these SVCs? Savvy, M-L (machine-language) programmers. No reason for them to spend hours figuring out the details of doing "sneaky" things, writing code for them, then debugging. Not if they can use someone else's handy routine already in RAM! Great! So, what if you're not an accomplished M-L programmer? Should you forget about SVCs? No, it's not quite THAT bad.

Note that each SVC requires the Z-80's A-Register to be loaded with, minimally, the function code and, often, other data has to be loaded into other registers as well. Then an "RST 8" command must be executed. Pre-loading the Z-80 Registers and executing an RST 8 are M-L tasks, and a tiny M-L program is needed to do them. Finally, upon return from the SVC, there may be information conveyed by the "C"-flag and/or the "Z"-flag and/or registers. Making use of that information is also a M-L program task. YOU CANNOT DO THESE CHORES WITHIN A BASIC PROGRAM!

If you are a M-L programmer, setting-up, using the RST 8, and going on about your business with the Z-80 are easy chores. If you are NOT a M-L programmer, you have three choices: (1) forget SVCs; (2) have someone else write the needed M-L program you can place in high Ram and then call it as needed from BASIC; or (3) learn to write tiny M-L programs and then proceed as in (2).

M- L Programs Brought To Life

Before you are tempted by the first choice, I have provided a "teaser" program which will allow anyone to use handy SVCs from BASIC. This small program can enable/disable the BREAK key (using SVC 3) and set CRT scroll-protection (using SVC 27).

```
                SVC/CMD
EF37 21 00 00 3E 03 CF C0 22   42 EF C9 FF FF 2A 42 EF
EF47 3E 03 CF C9 46 3E 1B CF   C9
```

The above are three little M-L programs in hex all run together! They don't look like much in "straight hex" M-L, do they?

Here's how we will make them come alive! First, make yourself a "clean" disk — a new one, formatted and backed-up from your TRSDOS BASIC working master, or a disk for which good backup copies are available for every file! Very strange things can happen in case of M-L errors. I wouldn't want you to lose a disk filled with treasured files if things went very wrong!

Second, put your fresh-clean disk in your Model II and bring it up to TRSDOS READY. Then, type

DEBUG ON (ENTER)

and type

DEBUG (ENTER)

You will see the top-half of the screen changed to the Debug Display. The flashing cursor will be on line 12 following a ? .

Press

(X)

just to be sure you are in hex addressing. Then press

(M)

and you will get A = ... asking for an address.

Type

EF37

The screen will change. The top address in the left-most column should read EF37.

Sixteen hex-number pairs immediately to the right indicate the current contents of the sixteen bytes of RAM that begin with hex address EF37.

Press

(F1)

The cursor will jump up onto the first one! Now type the twenty-five hex bytes EXACTLY as they appear above. The cursor will jump from one to the next an wrap-around to the next line automatically. Use the (arrow) keys to go back and make any corrections — all you have to do is type over. PROOFREAD VERY CAREFULLY!

On the second line of RAM following EF47 you should see 3E. In location EF4F there should be C9. Let's count over to it. The 3E following EF47 is in forty-seven. The 03 is in forty-eight. Then comes forty-nine, forty-A (Yes, A — we're in hex, remember!), forty-B, forty-C, forty-D, forty-E and, finally, forty-F. EF4F should be C9.

When you're SURE your entries are correct, press

```
(F2)
```

Next press

```
(ESC)
```

That should bring back the ?.

Press

```
(D)
```

and note the addresses change in the left column. You should see 61239 where the EF37 used to be: that's the decimal equivalent. Press

```
(X)
```

and hex addressing will return. Finally,

press

```
(S)
```

and you will return to TRSDOS READY.

The M-L routines are in RAM,and we don't want to ever type them in again! Type

```
DUMP SVCML/CMD START=EF37 END=EF4F RORT=R
(ENTER)
```

Careful, get it right! When TRSDOS READY is back, our M-L routines will be on disk in a file named SVCML/CMD. "RORT=R" simply tells TRSDOS not to try to execute the M-L program(s) after loading (disaster in this case!).

Next, type

```
DEBUG OFF (ENTER)
```

At TRSDOS READY type

```
BUILD SVC (ENTER)
```

Type

```
SVCML/CMD (ENTER)
```

and then

```
(ENTER)
```

on the first command line.

Type

```
BASIC SVCTEST/BAS -M:61239
(ENTER)
```

and then

```
(ENTER)
```

for the second command.

Just type

```
(ENTER)
```

for the next line to end the DO file. Later, NOT NOW, that will be handy. At TRSDOS READY, type

```
BASIC -M:61239 (ENTER)
```

After the BASIC Ready reply, type

```
CLS (ENTER)
```

Now, type in the BASIC program shown in LISTING 1.

Proofread carefully. BEFORE ANY ATTEMPT TO RUN, you should save the program to disk. You can accomplish this as follows:

Type

```
SAVE"SVCTEST/BAS" (ENTER)
```

Back at BASIC's Ready, type

```
RUN (ENTER)
```

The program should ask you to select which of the three routines you wish to test.

Press

(D) or (d) to disable the (BREAK) key
(E) or (e) to enable the (BREAK) key
(S) or (s) to set-up scroll protection for the 0 through 22 top lines of the screen.

Play with it!

If you have trouble, LIST and check carefully. If you can't find any error in the BASIC program, be sure you SAVE the latest version to the disk as above. Then type

```
SYSTEM (ENTER)
```

At TRSDOS READY, use DEBUG as above to inspect the M-L program. You may make any needed corrections and use DUMP again as before.

After turning DEBUG OFF — or anytime at TRSDOS READY you want to run my BASIC program (when DEBUG is off) — you need only type

DO SVC (ENTER)

The M-L and BASIC programs and BASIC itself with the proper high-RAM restriction (61239) will be loaded and a RUN will be given [unless you press the (BREAK) key immediately after the BASIC logo appears.]

NOTE: DEBUG must be off to use a DO file!*

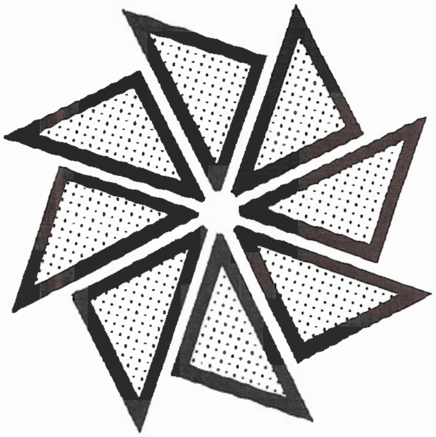
Let's Analyze The BASIC Program:
Line 10 defines the types of variables (I hate the suffix characters) and defines the handy code for cursor-off as string SF. Then it does CLS and turns off the cursor (which makes for "cleaner" displaying). The Remark is the disk-filespec of the program — as a reminder when you SAVE it.

Line 20 continues more single-byte codes to turn the blinking cursor on (SN), print on black background (SB), and print on white background (SW). Also included is a string list (SL) and a one-character-long string variable we'll use for input (SI). It's IMPORTANT SI be assigned as a one-character string; we'll use MID$(SI,1) later!

Line 30 assigns integers I=0 and X=0 and defines addresses (in hex) for USR0 through USR2 — so they can find the three little M-L routines they utilize. These addresses depend upon where the specific M-L routines are located in RAM.

While not strictly necessary for variables where the initial value is "don't care", assignment of all the variables to be used early in any program serves three purposes: (1) It's a handy reminder of the variable-names used (Have I used X1 ?); (2) It allows control of the order in which the variable-names are entered within the internal tables of the BASIC Interpreter (the further down, the longer the search); and (3) it minimizes the later relocation of the tables to accommodate new variable-names. It can be important to assign any DIM statements and/or any array-variables last!

Line 50 displays the "menu". Note the use of SW to obtain a black-on-white print!



Line 60 has SB to exit the "reverse" mode, as well as SN to turn the blinking-cursor on. The SF turns it off again. The statement MID$(SI,1)=INPUT$(1) may be new to you. With SI previously assigned as a one-character string, this allows input of one character from the keyboard WITHOUT BASIC CREATING ANY NEW ENTRY IN INTERNAL STRING-SPACE — helping avoid the dreaded "dead computer" during periodic "garbage collection".

Line 70 uses INSTR along with string SL to recognize valid input letters — upper or lower case. It's often simpler than the many IF SI="D" THEN.. , etc. statements usually seen for the purpose!

Line 80 is an error print and returns to line 50.

Line 100 gets down to the use of USR0. The variable X here is purely a matter of format; no data is passed-to nor returned-from the M-L routine in this case. (The value of X survives unchanged here.) What does happen is that BASIC passes control of the computer to the M-L program at address EF37 hex.

The M-L program there, in turn, sets up for and calls SVC function-code 3 to actually disable the processing of the (BREAK) key. Then, control returns to the M-L program which returns it to BASIC to interpret the next statement. That's a tabbed-over-to-31 reverse-mode print "(BREAK) DISABLED".. The SB returns the CRT to black-background mode.

Likewise, line 120 uses USR1 to enable the (BREAK) key. Both the lines continue with a printing demonstration in lines 140—170. When that follows line 100, you will be unable to (BREAK) the computer!

Line 190 inputs the number of lines to be scroll-protected.

Line 200 checks to be sure that number is in the range 0—22 or causes an error printout and return to line 190.

Line 210 has the critical X=USR2(X) statement. Here, a value is passed in the variable X which is used by the M-L program to properly set up for the SVC with Function Code 27. That actually does the scroll protection. No data passed back. The value of X survives unchanged — and is used in the reverse-mode print-out.

Lines 220 through 260 demonstrate scroll protection if any has been provided.

PLEASE NOTE THAT IF YOUR BASIC PROGRAM USES CLS OR PRINTS CHAR$(27), CHR$(30) or CHR$(31), THOSE AUTOMATICALLY CHANGE ANY SCROLL-PROTECTION TO ZERO LINES!

In my demonstration program I used CLS in line 10. Thus, each time you RUN the program, scroll-protection will be zero lines initially. However, at line 230, I did NOT use CLS! Instead, I used

PRINT@(0,0),CHR$(24);

to get the cursor to the upper-left and clear the screen. When a portion of the screen is scroll-protected, you may wish to clear only the unprotected portion. In that case, use the same trick but with the

*To "play with" my BASIC program with DEBUG operative, at TRSDOS READY type DEBUG ON and (ENTER) then SVCML/CMD and (ENTER) and BASIC SVCTEST/BAS -M:61239 and (ENTER). If you want to temporarily drop into DEBUG from BASIC's Ready, type SYSTEM"DEBUG" and (ENTER). To return to BASIC, be sure the PC-register contains 2800 in hex and press (C).

statement

PRINT@(r,0),CHR$(24);

where r is the number of lines you have scroll-protected.

My apologies for such a long BASIC program! Hopefully, you found enough tips in it to compensate for the typing.

Use Your Own BASIC Programs

To use these SVCs and the M-L programs in your own BASIC program, proceed as follows:

MOVE or COPY the file SVCML/CMD onto the disk you are using. BUILD a DO file similar to our SVC file but with the file-spec of your BASIC program instead of SVCTEST/BAS. (Keep the same RAM high-limit on BASIC at 61239.) Early in the program, assign the absolute addresses for USR functions as in my Line 30. Define an integer (MUST BE AN INTEGER!) such as X. Then, whenever you wish to disable the (BREAK) key, insert the statement X=USR0(X). Whenever you wish to enable the (BREAK) key, use X=USR1(X) . And whenever you wish to alter scroll-protection, (1) set X equal to the number of lines you want protected, and (2) use X=USR2(X) . That's all there is to it!

Well, almost.

Further Notes on Enabling/Disabling The Break Key

The M-L programs have built-in protection if you ask (BREAK) be enabled when it's already enabled or be disabled when already disabled. However, you should be careful to limit any input to the scroll-protect program to the range 0 through 22.

A BIG CAUTION ON USE OF (BREAK) KEY DISABLE: You can EASILY get a BASIC program into an inescapable loop with the (BREAK) key disabled! It can be disaster during check-out of the program — if you haven't SAVEed the most up-to-date version on disk. The only solution is the RESET switch. That will destroy the current RAM copy of the BASIC program. Even if you avoid endless loops, when you LIST you can temporarily stop the listing anywhere with the (HOLD) key BUT YOU WILL NOT BE ABLE TO (BREAK) THERE! Why disable the (BREAK) key at all? Too often in routine production use of a program, the operator presses the (BREAK) by accident. A touch typist goes right on typing! Finally, a look at the screen indicates something is very wrong. Then comes panic and the cry, "Oh! The computer's broken!" By then, it may even be too late (depending upon what was typed) to simply type CONT and press (ENTER). If the production version of the program had immediately disabled the (BREAK) key, then when a typist's quick finger strays over there, nothing catastrophic will happen.

My advice is to place the disabling statement on a separate line and insert a single quote in front so it's NEVER executed. When all program check-out is finished, then delete the prefix.

Another trick, useful in later program maintenance, is to conceal an extra valid input in the main-menu structure — not listed in the printed menu. Suppose, for example, our BASIC program had nothing to do with (BREAK) enable/disable.

Let's change line 20 so that the string-list is defined S=L "DdEeSs" + CHR$(126).

On line 70, add ,280 to the end of the ONIGOTO list.

At line 280, copy line 120 exactly, then add :GOTO50

Now, RUN the program. When anyone holds the (CTRL) key down while pressing the (6) as a menu selection, the (BREAK) key will be enabled and a confirming note will be displayed. Few users need ever know that such a feature exists! You might wish to conceal such a "secret key" in every menu in a large program.

FINALLY, ALWAYS ENABLE THE (BREAK) KEY BEFORE ENDING ANY OF YOUR PROGRAMS IN WHICH YOU DISABLE IT!

A Few Closing Comments

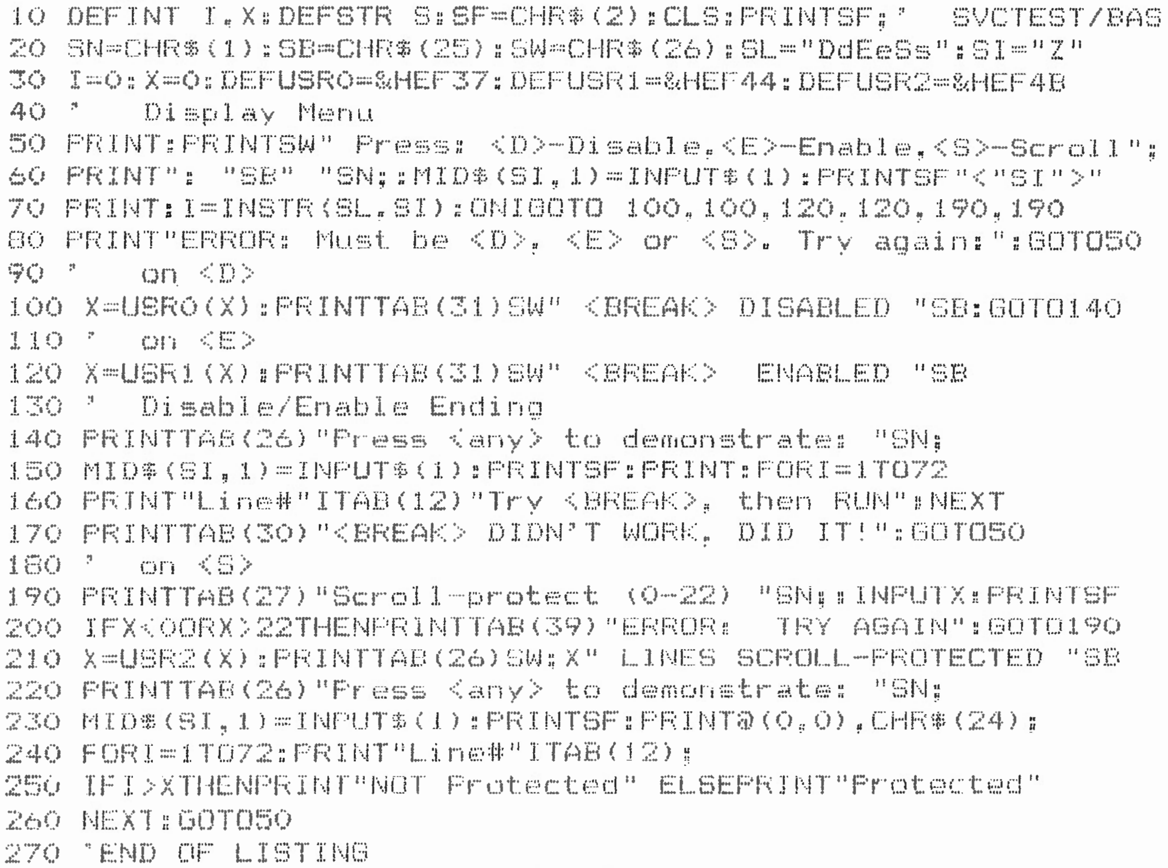
It's possible to write a M-L program that can serve as a Universal SVC-Driver allowing a BASIC program to utilize ANY of the SVCs. However, it won't be as efficient and will require much more understanding by the BASIC programmer.

Normally, I use RAM above hex-address F000 for M-L programs allowing full use of all normal BASIC RAM area. The default when no limit is stated on loading BASIC is -M:61439 or EFFF hex — the byte before F000 hex.

Here I've used a lower limit, depriving BASIC of two-hundred bytes for two reasons: (1) If readers use SERIAL output, the serial comm-drivers are likely in the F000 through F3FF area of RAM. (Parallel printers don't need those.) (2) I wanted to give an illustration of the procedure for limiting BASIC's use of RAM. The allocation of two hundred bytes leaves room for additional M-L programs to be added to your "library" later.

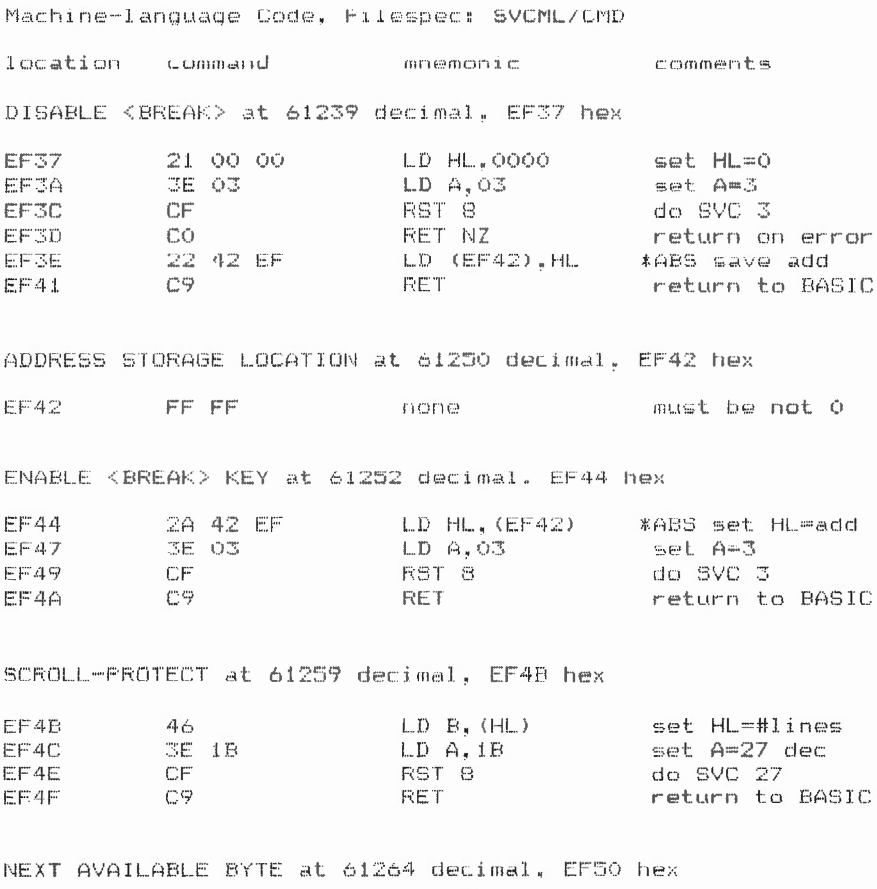
The actual M-L programs used here (Listing Two) are in no particular assembler format as I didn't use one. Write M-L without an assembler? Sure, it's not THAT difficult!

Turn the page for Listing One (the BASIC program) and Listing Two.



```
10 DEFINT I,X:DEFSTR S:SF=CHR$(2):CLS:PRINTSF;'   SVCTEST/BAS
20 SN=CHR$(1):SB=CHR$(25):SW=CHR$(26):SL="DdEeSs":SI="Z"
30 I=0:X=0:DEFUSR0=&HEF37:DEFUSR1=&HEF44:DEFUSR2=&HEF4B
40 '    Display Menu
50 PRINT:PRINTSW" Press: <D>-Disable,<E>-Enable,<S>-Scroll";
60 PRINT": "SB" "SN;:MID$(SI,1)=INPUT$(1):PRINTSF"<"SI">"
70 PRINT:I=INSTR(SL,SI):ONIGOTO 100,100,120,120,190,190
80 PRINT"ERROR: Must be <D>, <E> or <S>. Try again:":GOTO50
90 '    on <D>
100 X=USR0(X):PRINTTAB(31)SW" <BREAK> DISABLED "SB:GOTO140
110 '   on <E>
120 X=USR1(X):PRINTTAB(31)SW" <BREAK>  ENABLED "SB
130 '   Disable/Enable Ending
140 PRINTTAB(26)"Press <any> to demonstrate: "SN;
150 MID$(SI,1)=INPUT$(1):PRINTSF:PRINT:FORI=1TO72
160 PRINT"Line#"ITAB(12)"Try <BREAK>, then RUN":NEXT
170 PRINTTAB(30)"<BREAK> DIDN'T WORK, DID IT!":GOTO50
180 '   on <S>
190 PRINTTAB(27)"Scroll-protect (0-22) "SN;:INPUTX:PRINTSF
200 IFX<0ORX>22THENPRINTTAB(39)"ERROR:  TRY AGAIN":GOTO190
210 X=USR2(X):PRINTTAB(26)SW;X" LINES SCROLL-PROTECTED "SB
220 PRINTTAB(26)"Press <any> to demonstrate: "SN;
230 MID$(SI,1)=INPUT$(1):PRINTSF:PRINT@(0,0),CHR$(24);
240 FORI=1TO72:PRINT"Line#"ITAB(12);
250 IFI>XTHENPRINT"NOT Protected" ELSEPRINT"Protected"
260 NEXT:GOTO50
270 'END OF LISTING
```

Listing One



```
Machine-language Code, Filespec: SVCML/CMD

location    command          mnemonic          comments

DISABLE <BREAK> at 61239 decimal, EF37 hex

EF37        21 00 00         LD HL,0000        set HL=0
EF3A        3E 03            LD A,03           set A=3
EF3C        CF               RST 8             do SVC 3
EF3D        C0               RET NZ            return on error
EF3E        22 42 EF         LD (EF42),HL     *ABS save add
EF41        C9               RET               return to BASIC


ADDRESS STORAGE LOCATION at 61250 decimal, EF42 hex

EF42        FF FF            none              must be not 0


ENABLE <BREAK> KEY at 61252 decimal, EF44 hex

EF44        2A 42 EF         LD HL,(EF42)     *ABS set HL=add
EF47        3E 03            LD A,03           set A=3
EF49        CF               RST 8             do SVC 3
EF4A        C9               RET               return to BASIC


SCROLL-PROTECT at 61259 decimal, EF4B hex

EF4B        46               LD B,(HL)         set HL=#lines
EF4C        3E 1B            LD A,1B           set A=27 dec
EF4E        CF               RST 8             do SVC 27
EF4F        C9               RET               return to BASIC


NEXT AVAILABLE BYTE at 61264 decimal, EF50 hex
```

Listing Two

USING TRSDOS SUPERVISOR CALLS

Dr. Gary Turkel
137 Lafayette Ave.
Suffern, NY 10901

The TRSDOS Supervisor Calls (SVCs) are convenient and efficient aids for accomplishing many programming chores. You should think of them as a library of subroutines to handle numerous functions: input-output (video, keyboard printer, and disk), searches, sorts, and computations. SVCs allow you, as the programmer, to do many operating system chores without having to return the TRSDOS READY command level and without involving the user.

The SVCs can be called only within an assembly language program. This assembly language program can then be called from a high level language (e.g. BASIC, COBOL), or from another assembly language program, or from the TRSDOS READY command level. My rule of thumb limits assembly language program usage to those tasks that cannot be performed by a high level language, in my case COBOL.

SVCs can be tricky because some features of an SVC may be nullified by certain TRSDOS characteristics we don't know about. This can make testing and using an SVC very difficult.

I remember the first SVC I tried to use: SCROLL. This Supervisor Call protects a number of upper screen lines from being scrolled off the screen. I spent many unsuccessful hours trying to make it work. Finally, in desperation, I decided to use it in my program anyway to see what would happen. Sure enough, it worked! I had been testing it from the TRSDOS command level, not realizing that whenever TRSDOS READY appeared, the number of lines to be protected had been re-initialized to zero. Give Tandy a C for their documentation!! And the only reason they get a grade that high is because they are better than all others I've seen.

SVCs And Z80 Programming

Since calling an SVC requires an assembly language routine, you need at least a rudimentary understanding of Z80 code. While such a tutorial is not within the scope of this article, I will attempt to explain those features essential to writing a Z80 program that will call an SVC.

The Z80 CPU consists of, among other things, seven general purpose registers (contiguous memory cells) and a special register called the Flag register. (Other registers and components are not needed for the SVCs and are not included in this discussion.) Each register has a letter name: A,B,C,D,E,F,H, and L. Certain registers may be grouped together to form register pairs: AF,BC,DE, and HL. The Z80 gets all its data from memory locations (addresses) named numerically from zero through 65535.

Each of the above addresses and registers consists of eight memory cells, with each cell holding a binary digit called a bit; eight bits make a byte. Numbers from 256 through 65535 require more than eight bits for binary representation, hence the need for a register pair (two adjacent memory locations) to hold numbers in this range. Note that the register pair AF cannot, however, be used for this purpose.

There are two types of data: character data and numeric data. Character data are letters, punctuation symbols, numeric symbols, and graphics characters which are represented by an eight bit code called the ASCII code. Numeric data may take many forms. Three that relate to this discussion are addresses (require sixteen bits), integers (may be eight or sixteen bits), and Opcodes (may be eight or sixteen bits). Opcodes are numeric instructions for the micro-processor. They may be combined with sixteen bits of address data, eight bits of character data, or an eight or sixteen bit integer to form a complete instruction.

SVC program instructions fall into two categories: (1) those that transfer data from one place to another, and (2) those that alter the memory location from which the Z80 will get its next instruction. Data may be transferred internally in the microprocessor from register to register, or it may be transferred between a register and a memory location. Normally, the Z80 will get its instructions from the next sequential location in memory. However, a number of instructions will tell the CPU to get its next instruction from someplace other than the next memory location.

SVC 27: SCROLL

Let's go through a program that will call the SVC SCROLL (function code 27). SCROLL is written in Z80 mnemonic code (mnemonic code is a symbolic representation of the actual numbers of an instruction). The mnemonics are converted into actual numbers by a program called an assembler. (Those without an assembler can still create a program file by entering the instruction into memory using the DEBUG utility. See sidebar.)

SCROLL protects a designated number of upper screen lines from being scrolled off the screen. The screen is scrolled when a line feed is sent to the screen, and the cursor is on the 24th line. A line feed is sent to the screen whenever the operating system displays a character in the 80th column of the screen (or it can be done using the SVC VDCHAR [function code 8]). The parameters, or "Entry Conditions" as *The Owner's Manual* calls them, required for SCROLL tell us that before calling the SVC, the B register must contain the number of upper screen lines to be protected (zero through 22 are the only valid numbers). The A register must contain the number 27, which is the function code number for this SVC.

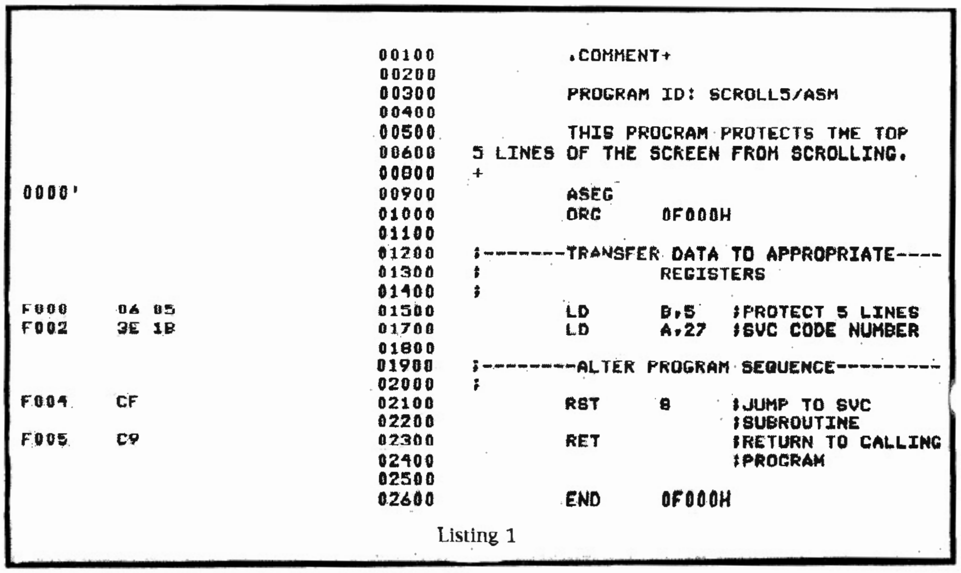
Now to the program in Listing 1. The mnemonics in the 9th and 10th lines tell the assembler the memory location for the start of the program — F000 hexadecimal. The numbers in the columns to the left of the line numbers are in hexadecimal format generated by the assembler. The column on the far left contains the addresses of the first byte of each instruction. The next columns contain the instructions, with each two digit hex number representing a byte of code. From this, we know that the instruction at address F000 has the hex value 06, and the contents of the next address, F001, is a number with a hex value of 05.

The first program instruction is located on the 15th line; it tells the Z80 to transfer (LD) the number 5 to register B. The next line instructs the CPU to transfer the number 27 (1Bhex) into the A register.

At this point, we're ready to call the SVC by using the instruction on the 21st line. RST 8 is an instruction that tells the CPU to get its next instruction from memory location number 8. When it finishes the instructions there, it will be ready to continue with the instruction immediately following the RST 8.

IMPORTANT: The RST 8 instruction is THE method of calling ALL SVCs. The code beginning at address number 8 will decode the SVC code number (in register A), execute the SVC, and then return the CPU to the address following the RST 8 instruction for its next instruction. In Listing 1, this next instruction is a RET (return) that tells the CPU to go back to the program it was executing before it was directed to this one.

Something to remember: a return to TRSDOS READY, a CLS statement in BASIC, or an ERASE option of the

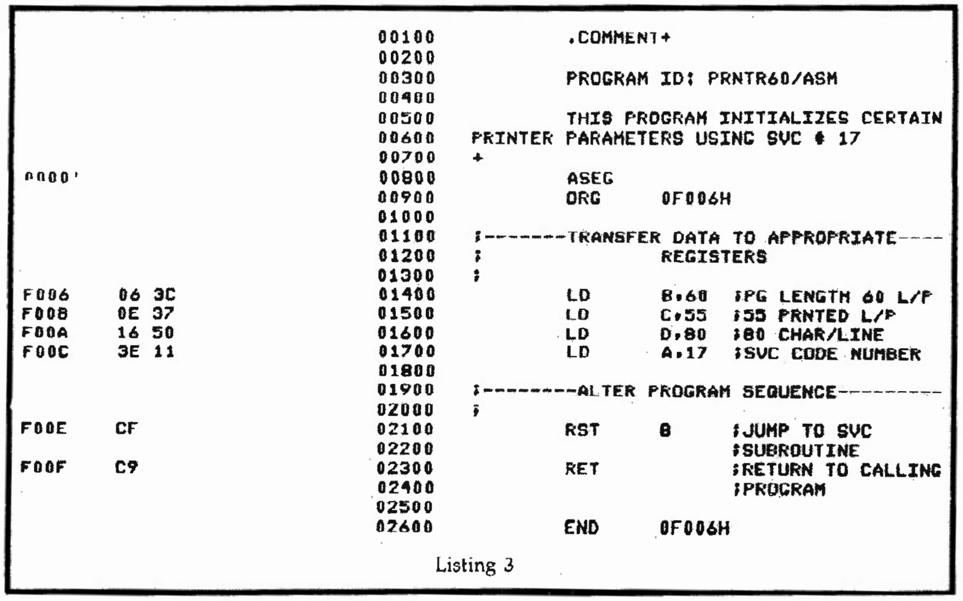


```
                           00100          .COMMENT+
                           00200
                           00300          PROGRAM ID: SCROLL5/ASM
                           00400
                           00500          THIS PROGRAM PROTECTS THE TOP
                           00600    5 LINES OF THE SCREEN FROM SCROLLING.
                           00800    +
0000'                      00900          ASEG
                           01000          ORG       0F000H
                           01100
                           01200    ;-------TRANSFER DATA TO APPROPRIATE----
                           01300    ;                 REGISTERS
                           01400    ;
F000      06 05            01500          LD        B,5      ;PROTECT 5 LINES
F002      3E 1B            01700          LD        A,27     ;SVC CODE NUMBER
                           01800
                           01900    ;--------ALTER PROGRAM SEQUENCE---------
                           02000    ;
F004      CF               02100          RST       8        ;JUMP TO SVC
                           02200                             ;SUBROUTINE
F005      C9               02300          RET                ;RETURN TO CALLING
                           02400                             ;PROGRAM
                           02500
                           02600          END       0F000H
```

Listing 1

DISPLAY statement in COBOL are some of the ways that the number of protected lines is re-initialized to zero.

Of course, this SVC program is useless by itself; it must interact with another program designed to list items onto the screen. A sample COBOL program demonstrating the use of the SCROLL/ASM program is shown in Listing 2. Notice that the assembly language program is called AFTER the screen is cleared. In BASIC, the assembly language program is called after each CLS command.



```
                                  00100          .COMMENT+
                                  00200
                                  00300          PROGRAM ID: PRNTR60/ASM
                                  00400
                                  00500          THIS PROGRAM INITIALIZES CERTAIN
                                  00600   PRINTER PARAMETERS USING SVC # 17
                                  00700   +
0000'                             00800          ASEG
                                  00900          ORG      0F006H
                                  01000
                                  01100   ;-------TRANSFER DATA TO APPROPRIATE----
                                  01200   ;               REGISTERS
                                  01300   ;
F006      06 3C                   01400          LD       B,60    ;PG LENGTH 60 L/P
F008      0E 37                   01500          LD       C,55    ;55 PRNTED L/P
F00A      16 50                   01600          LD       D,80    ;80 CHAR/LINE
F00C      3E 11                   01700          LD       A,17    ;SVC CODE NUMBER
                                  01800
                                  01900   ;--------ALTER PROGRAM SEQUENCE---------
                                  02000   ;
F00E      CF                      02100          RST      8       ;JUMP TO SVC
                                  02200                           ;SUBROUTINE
F00F      C9                      02300          RET              ;RETURN TO CALLING
                                  02400                           ;PROGRAM
                                  02500
                                  02600          END      0F006H
```

Listing 3

More Uses For SVCs: SVC -17 PRINIT

I have two objectives I keep before me during my programming applications. One is to avoid exiting the program and returning to the TRSDOS READY command level to do an operating system chore; the other is to save the operator from having to perform any operating system chore. The next two programs demonstrate the methods I use to achieve these objectives using the SVCs.

The first program, utilizing SVC 17 PRINIT (Listing 3), is a beautiful substitute for the FORMS command. For one thing, the printer need not be turned on to execute it. For another, its usage eliminates the need for the operator to know and type in the FORMS command when needed. All you have to do is make up one of the programs incorporating PRINIT for each size form or report application you have and call it before you start printing.

Even better, you can accomplish the same things with only one program if you know a little fancy programming. Pass the values for the printer parameters from your high level program to your SVC program. That's what I did. TRSDOS initializes the printer parameters to an inconvenient set of values for my printer on boot-up, so I inserted one of these programs in the DO file that is automatically executed on boot-up.

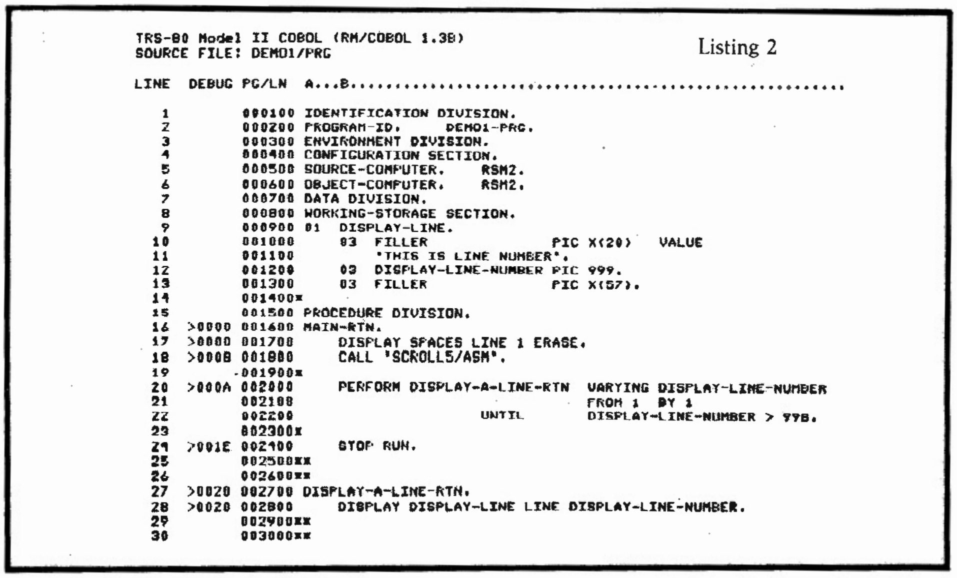
When using the SVC PRINIT, Register A must contain the number 17, the SVC function code number. Page length, printed lines per page, and characters per line must be placed in registers B, C, and D respectively before the SVC is called with the RST 8 instruction.

For the uninitiated, page length is the maximum number of lines that can be printed on a page; printed lines per page is the number of lines that may be printed before going to the next page; characters per line is the number of print positions on each line and is usually set to either 80 or 132, depending on the report to be generated or the printer size.

Try a number of programs calling PRINIT, and experiment by varying the values of registers B, C, and D each time. Note: Register B can never contain a value less than that contained in Register C.

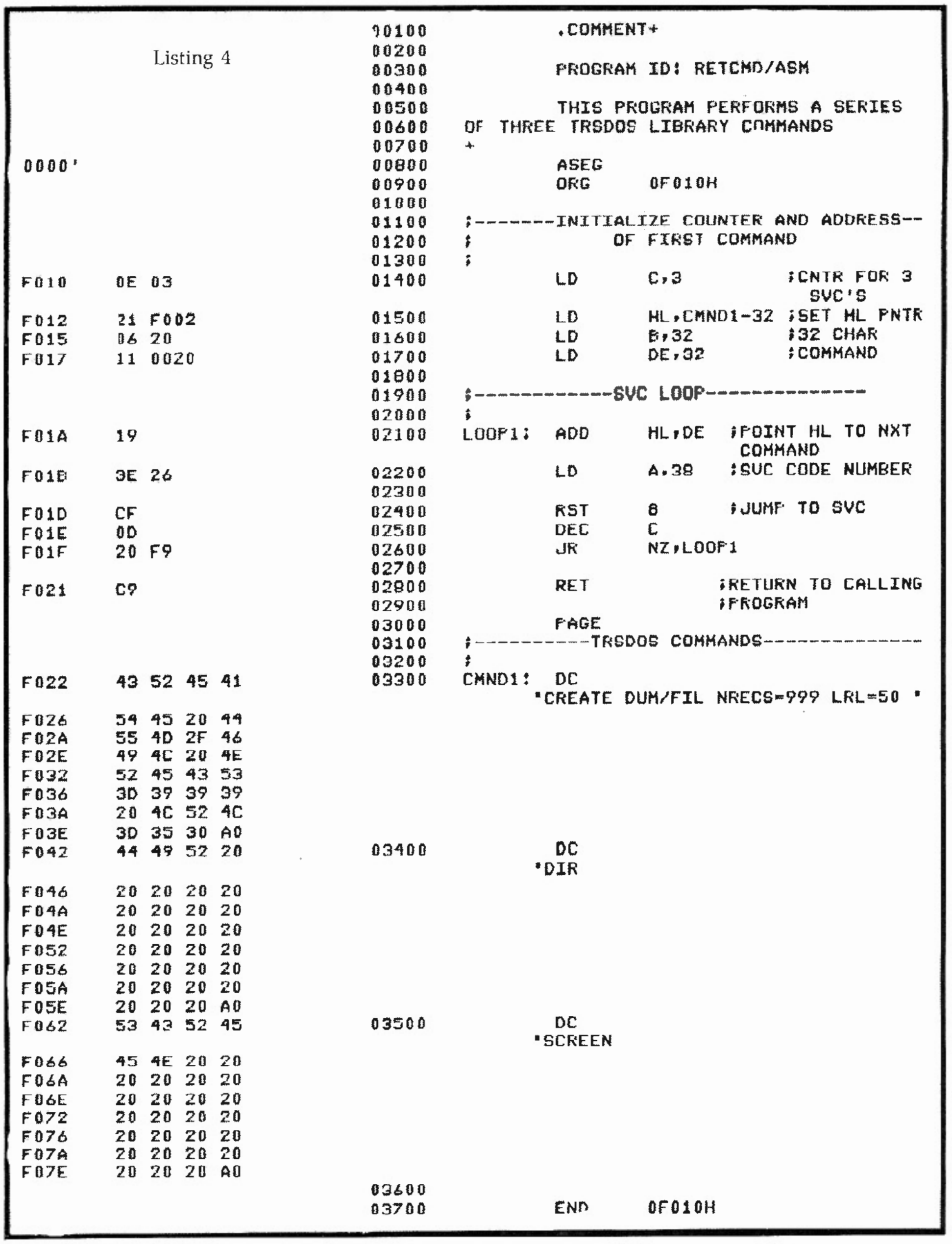
SVC -38 RETCMD

My final program uses SVC 38 RETCMD (function code 38) (see Listing 4). When I found it, I thought I might have discovered *the ultimate* SVC. However, I soon turned up some problems associated with its usage. The SVC RETCMD executes a TRSDOS library



Listing 2

```
TRS-80 Model II COBOL (RM/COBOL 1.3B)
SOURCE FILE: DEMO1/PRG

LINE   DEBUG PG/LN  A...B.......................................................

  1          000100 IDENTIFICATION DIVISION.
  2          000200 PROGRAM-ID.      DEMO1-PRG.
  3          000300 ENVIRONMENT DIVISION.
  4          000400 CONFIGURATION SECTION.
  5          000500 SOURCE-COMPUTER.    RSM2.
  6          000600 OBJECT-COMPUTER.    RSM2.
  7          000700 DATA DIVISION.
  8          000800 WORKING-STORAGE SECTION.
  9          000900 01  DISPLAY-LINE.
 10          001000      03  FILLER                PIC X(20)   VALUE
 11          001100          'THIS IS LINE NUMBER'.
 12          001200      03  DISPLAY-LINE-NUMBER PIC 999.
 13          001300      03  FILLER                PIC X(57).
 14          001400*
 15          001500 PROCEDURE DIVISION.
 16    >0000 001600 MAIN-RTN.
 17    >0000 001700      DISPLAY SPACES LINE 1 ERASE.
 18    >000B 001800      CALL 'SCROLL5/ASM'.
 19          -001900*
 20    >000A 002000      PERFORM DISPLAY-A-LINE-RTN  VARYING DISPLAY-LINE-NUMBER
 21          002100                                  FROM 1  BY 1
 22          002200                        UNTIL      DISPLAY-LINE-NUMBER > 998.
 23          002300*
 24    >001E 002400      STOP RUN.
 25          002500**
 26          002600**
 27    >0020 002700 DISPLAY-A-LINE-RTN.
 28    >0020 002800      DISPLAY DISPLAY-LINE LINE DISPLAY-LINE-NUMBER.
 29          002900**
 30          003000**
```



Listing 4

```
                     00100          .COMMENT+
                     00200
                     00300          PROGRAM ID: RETCMD/ASM
                     00400
                     00500          THIS PROGRAM PERFORMS A SERIES
                     00600  OF THREE TRSDOS LIBRARY COMMANDS
                     00700  +
0000'                00800          ASEG
                     00900          ORG     0F010H
                     01000
                     01100  ;-------INITIALIZE COUNTER AND ADDRESS--
                     01200  ;               OF FIRST COMMAND
                     01300  ;
F010    0E 03        01400          LD      C,3            ;CNTR FOR 3
                                                            SVC'S
F012    21 F002      01500          LD      HL,CMND1-32 ;SET HL PNTR
F015    06 20        01600          LD      B,32           ;32 CHAR
F017    11 0020      01700          LD      DE,32          ;COMMAND
                     01800
                     01900  ;------------SVC LOOP-------------
                     02000  ;
F01A    19           02100  LOOF1:  ADD     HL,DE   ;POINT HL TO NXT
                                                     COMMAND
F01B    3E 26        02200          LD      A.38    ;SUC CODE NUMBER
                     02300
F01D    CF           02400          RST     8       ;JUMF TO SVC
F01E    0D           02500          DEC     C
F01F    20 F9        02600          JR      NZ,LOOF1
                     02700
F021    C9           02800          RET                  ;RETURN TO CALLING
                     02900                               ;FROGRAM
                     03000          FAGE
                     03100  ;----------TRSDOS COMMANDS-------------
                     03200  ;
F022    43 52 45 41  03300  CMND1:  DC
                                  'CREATE DUM/FIL NRECS=999 LRL=50 '
F026    54 45 20 44
F02A    55 4D 2F 46
F02E    49 4C 20 4E
F032    52 45 43 53
F036    3D 39 39 39
F03A    20 4C 52 4C
F03E    3D 35 30 A0
F042    44 49 52 20  03400          DC
                                  'DIR
F046    20 20 20 20
F04A    20 20 20 20
F04E    20 20 20 20
F052    20 20 20 20
F056    20 20 20 20
F05A    20 20 20 20
F05E    20 20 20 A0
F062    53 43 52 45  03500          DC
                                  'SCREEN
F066    45 4E 20 20
F06A    20 20 20 20
F06E    20 20 20 20
F072    20 20 20 20
F076    20 20 20 20
F07A    20 20 20 20
F07E    20 20 20 A0
                     03600
                     03700          END     0F010H
```

command and then returns to your program for the next instruction. This is very nice, but it doesn't eliminate operator interaction for those commands usually requiring it (e.g., KILL, FORMS).

Another of RETCMD's shortcomings involves the limits imposed on its application: some of its most useful commands are not callable from BASIC or COBOL because they overlay areas of memory also used by BASIC and COBOL (e.g., CREATE, KILL, COPY, MOVE). A COPY or MOVE can usually be accomplished from within a high level language, and there is an SVC for KILL (function code 41, named, appropriately, KILL) so the problem is not insurmountable. However, CREATE does create a problem. Although SVC OPEN (function code 40) is able to create a file, creating a file is only a sidelight to OPEN's main purpose, which is to open a file. Thus, there is no way to control the size of the pre-allocated file. If file size is not important, an OPEN statement will create a file of minimal size.

A second-rate solution I devised to get around this limitation is to use the RETCMD SVC to create a dummy file of the size that might be needed later, then use the DOSCMD SVC (function code 37) to call the main high level language program. When the name and size of the needed file are known, the RENAME SVC (code 47) could be used in a program called from the high level language. This wastes a lot of disk space, and there might not even be enough space, but occasionally compromises are unavoidable. Perhaps some day Tandy will give us a decent SVC for creating a file.

Getting back to the program using RETCMD SVC (Listing 4), the Entry Conditions, found in RETCMD's specifications in *The Owner's Manual*, have been complied with: Register A contains the SVC code 38, and the B register contains the length of the command string in bytes. The paired letters "HL" in Line 15 mean the HL register pair must contain the address of the first byte of the command string. The instruction on line 21 of the program accomplishes this requirement. I have made the second and third strings longer than necessary but the same length as the first string to simplify the programming.

Peculiarities Of The Microsoft Assembler

I must digress for a moment to point out two oddities of the Microsoft (licensed to Radio Shack) assembler. The DC statement tells the assembler to convert the character string between the quote marks to a series of ASCII numbers to be placed in memory starting at the current location in the program. Microsoft, for reasons unknown to me, decided that the last byte of the string should have its most significant bit set to one. This puts the value of the last byte outside the range of ASCII values.

In the program shown in Listing 4, the RETCMD SVC is expecting only ASCII code in the string. Because of this, it ignores the last byte of the string. This is a critical error for the first command string! The Microsoft assembler changed the last value in the string from 50 to 5. The solution I used here to counteract such a devastating error was to add another byte to the string.

The second — and in my opinion, more significant — fault with the Microsoft Assembler is the incorrect printing of two byte numbers in the third column of the assembler listing. On the 15th line of Listing 4, the hex number F003 appears. Although this is a valid hex address, it does not correctly represent the contents of the memory locations intended. It should be printed as ' 03 F0 ' indicating that the contents of address F013 are 03hex and the contents of address F014 are F0hex.

This reversal must be remembered if you are using Listing 4 to enter the program in memory using the DEBUG utility. Note also that the 17th line contains a similar reversal error. The instructions on this line should be printed as ' 11 1F 00 '.

The Question Of Error Routines

I have so far ignored the subject of errors. Strict programming practice would indicate that these programs be able to handle an error condition. Personally, I don't include any error routines in my programs. I feel they have such narrow functions that thorough testing will reveal any programming errors. Additionally, error routines will usually take up more memory space than the main program. This space is in very short supply indeed when you consider that all the space below F000 must be reserved for TRSDOS and the high level program, and the space from F300 to FFFF must at least be shared with TRSDOS and video memory.

For those who want to write error routines, Register A will contain an error code when the CPU returns to your program for the next instruction after the RST 8 instruction. A list of the codes and their meaning can be found in *The Owner's Manual.*

A Final Word For BASIC Users

BASIC programmers should note that many of the functions of the SVCs can be accomplished using existing BASIC statements or keywords. CHR$, INKEY$, HEX$, LOC, and DATE$ are a few that come to mind. COBOL, unfortunately, is more limited in this respect. There are many SVCs that will enhance programs in either language.

If you don't have an assembler, you can create a program file utilizing SVC 27 SCROLL or any other assembly language program by hand assembling, using a Z80 Opcode listing to get the hex numbers of the mnemonics and a decimal-to-hex conversion table to convert the data to hex format.

For the program in Listing 1, the following steps should be followed:

From TRSDOS READY type in

DEBUG ON (ENTER)

type in

DEBUG (ENTER)

This puts you in the DEBUG utility.

Press the following keys (DON'T PRESS ENTER):

(M) (F1 key) (0) (0) (0) (F1) (0) (6) (0) (5) (3) (E) (1) (B) (C) (F) (C) (9) (F2 key) (O) (this is the letter O)

Now the instructions are in memory in hex format. Note that the digits placed in memory between the (F1) key and the (F2) key are the same sequence of digits as in columns 2 and 3 of Listing 1.

```
TRS-80 Model II DEBUG Program
F000     06 05 3E 1B CF C9 FF 00   FF 00 FF 00 FF 00 FF 00    ..>.............
F010     FF 00 FF 00 FF 00 FF 00   FF 00 FF 00 FF 00 FF 00    ................
F020     FF 00 FF 00 FF 00 FF 00   FF 00 FF 00 FF 00 FF 00    ................
F030     FF 00 FF 00 FF 00 FF 00   FF 00 FF 00 FF 00 FF 00    ................
F040     FF 00 FF 00 FF 00 FF 00   FF 00 FF 00 FF 00 FF 00    ................
F050     FF 00 FF 00 FF 00 FF 00   FF 00 FF 00 FF 00 FF 00    ................
F060     FF 00 FF 00 FF 00 FF 00   FF 00 FF 00 FF 00 FF 00    ................
F070     FF 00 FF 00 FF 00 FF 00   FF 00 FF 00 FF 00 FF 00    ................
  PC    SP   SZHPNC  AF   BC   DE   HL   IX   IY   AF'  BC'  DE'  HL'
 2800  21FE  000000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
? O
DEBUG is now ON
TRSDOS READY
DEBUG
DEBUG is now OFF
TRSDOS READY
DUMP SCROLL3/ASM START=F000 END=F005
TRSDOS READY
SCREEN
```

Listing 5

Call or Write Your Nearest Snappware Distributor

MICRO-80
284 Goodwood Road
Clarence Park
Adelaide South Australia
Ph-(08) 2117224

DIGI-TEK SYSTEMS
65 Thornridge Circle
Kitchener Ontario N2M-4V9
Ph-(519) 742-8205

STRAWFLOWER ELECTRONICS
50 North Cabrillo
Half Moon Bay
California 94019
Ph-(415) 726-9128

COMPUTER MAGIC
115 Wiltshire Avenue
Louisville, Kentucky 40207
Ph-(502) 893-9334

E-C DATA
Tornevangsvej 88
P.O.B. 116
DK-3460
Birkerod, Denmark
Ph +45/2/81/ 81 91

SYSTEM SOFT
49, Dunvegan Drive
Rise Park
Nottingham, England
NG5 5Dx
Ph-(0602) 275559

CUSTOM COMPUTING
104 Bushwick Rd.
Poughkeepsie, NY 12603
Ph-(914)-471-9318

AEROCOMP
Redbird Airport
Hanger 8
Dallas, Texas 75232
Ph-(214) 339-5104

NOTICE

Snappware Goes On Trial! You Be The Judge!

Snappware knows your programming time is valuable. That's why we are offering a trial package that will cut your programming time up to 75%!

This unprecedented offer allows you to judge for yourself the value of our software using *your* hardware at *your* convenience. Our five best selling products: EXTENDED BASIC, EXTENDED BUILT IN FUNCTIONS, COLLEGE EDUCATED GARBAGE COLLECTOR, AUTOMAP and AUTOFILE, are all available to you on a trial basis for only $50.00 for the Model II and $35.00 for the Model III. We're convinced that after you see how well our software helps you perform programming tasks, you will purchase them like thousands of others have.

Our trial package consists of a master diskette which may be used to create one working copy. Your purchase price for the trial package will be credited toward the purchase of any software Snappware sells.

If for any reason you are not satisfied, just return the trial diskette and working copy and we will refund your money, no questions asked. With an unconditional guarantee like this, you can't lose.

Call our toll free number 800-543-4628 to put us on trial. We're sure your verdict will be: Snappware saves you time and money.

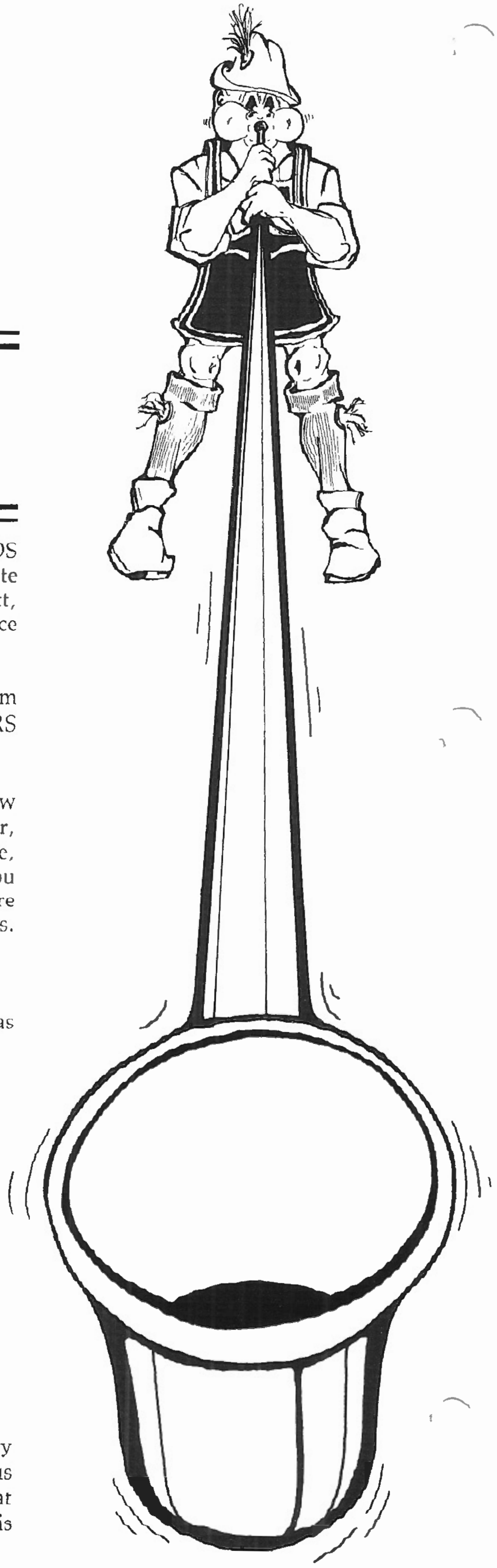
MODEL II Trial Package	**......$50.00**
MODEL III Trial Package	**......$35.00**

SNAPPWARE
SNAPPWARE
SNAPPWARE
SNAPPWARE

Time saving power at your fingertips.
CALL TOLL FREE:
1-800-543-4628
OHIO RESIDENTS CALL COLLECT: (513) 891-4496
3719 Mantell
Cinti., Ohio 45236

HOW TO USE THE SVCs ON THE MODEL II

Roger Conant
University of Illinois
Circle Campus
Box 4348
Chicago, IL 60680



SuperVisor Calls — or SVCs as they are most commonly called — are TRSDOS machine language subroutines that are always stored in memory. They are used quite often by TRSDOS itself, but it's possible for you to use them directly, too. In fact, it's hard to imagine doing much assembly language programming without them since they make it so easy to do.

Although SVCs are primarily for assembly language programmers, you can use them without being an "assembler freak." And you can use them without having the TRS assembler disk and programs. All you need is DEBUG.

This article will show you how to use the SVCs. I start out by assuming you know practically nothing, even about DEBUG. I make the further assumption, however, that you are a fast learner! So by the time you are half way through this article, you will have used the SVCs to enter and save a simple program (one which you might even find useful). After passing the midway point, the article gets into more technical information intended primarily for use by assembly language programmers.

A Few Hands-On Demonstrations (With Explanatory Patter)

Before we get started, the symbols I've used throughout this article are defined as follows:

* (asterisk) represents "Press the ENTER key"
[F1] represents "Press the F1 key"
[F2] represents "Press the F2 key"

To run your first SVC, rev up your Model II and type in:

CLS* (Remember, the "*" stands for "Enter")

CLEAR*

DEBUG ON*

DEBUG*

Entering these few commands will display the machine memory. You'll see the memory for hex addresses from 2800 to 287F, and you'll see displays of the contents of various registers (PC SP, SZHPNC, AF, BC, DE, HL, IX, IY, AF', BC', DE', HL'). Note that the contents of the A Register (the left half of "AF") are zero. In the vernacular, this means that the A Register is "Loaded" with 0.

In DEBUG you practically never type "Enter;" note the absence of "*" in what follows. Now type in the things shown on the left side below:

M	This enters "Memory" mode of DEBUG. The machine will type "A="; it's asking for an address
[F1]	The cursor jumps to cell 2800 and waits for your data
CFC7	Here you are entering a very small program into the memory; it will "call" the SVC subroutine and then exit to TRSDOS
[F2]	This saves your program in memory. If you have made a mistake, start over again with the "M"
C	This means "Run the Program"

If you've done everything correctly, you'll hear the disk whirr, leave DEBUG, and end up back at TRSDOS READY. Actually, you have just run a program using SVC #0 INITIO that initializes all I/O devices. OK, so it wasn't very exciting, but it was a start.

The next one is more fun. Type in:

DEBUG*	You get the display again, and your "CFC7" is still there
R	This puts you into the "Register" mode of DEBUG. The machine asks "=>" meaning, "Which register should I alter?"
A	Cursor jumps to the A register display
08	This loads A with zero eight, and the zero is very important! Loading A with 08 means that SVC #8 will be used
>>	Use the right arrow key to space over two spaces to the B register
1A	Load B with 1A hex (=26 decimal)
[F2]	This saves the contents of the registers
C	Run the program

There! Now that really produced something noticeable! What you did was invoke SVC #8 VDCHAR whose function is to display one character on the screen. And you did this by establishing the special "Entry Conditions" for SVC #8.

Entry Conditions supply instructions to TRSDOS by telling it which SVC subroutine to use. And the Entry Conditions also tell the subroutine which data to use. Each of the various SVCs requires its own special Entry 2Conditions.

A = 8 is the required Entry Condition for SVC #8, and following the placement of this value in Register A, certain required data must be placed in Register B. For SVC #8 the required data is the ASCII code of the character — 1A hex or 26 decimal, a "control" character whose function is to turn on the reverse video.

More on the ATON/WordStar Issue

(Continued from page 9)

Ed. note: Just as this issue of the magazine was ready to go to press, Chuck Wilde, President of ATON, called to offer a few words concerning the installation of WordStar with ATON CP/M. He told us that the ATON people have been successful in their efforts to get WordStar running with ATON's version of CP/M without the lock-ups experienced by our reviewer. The key to a successful ATON/WordStar installation and operation lies in using a "clean copy" of the installation command program WSU.COM. If WordStar has been installed to run under a Pickles & Trout or Lifeboat CP/M prior to its being installed on ATON, certain idiosyncracies inherent in the other operating systems appear to carry over to the ATON installation. By using a "clean, never-used-before copy" of WSU.COM with either the SOROC 120 terminal or the Lier-Siegler ADM 3 terminal, WordStar will be up and running under ATON CP/M with no problem.

You can repeat the above experiment with any of the 128 ASCII characters (00 hex through FF hex). For example, to get rid of that unpleasant reverse video, type:

DEBUG*	You should find your program (CFC7) still there
R	Machine asks "=> "
A	Getting ready to load A again
08	Load A with 08
>> 19	Load B with 19 hex (=25 decimal)
[F2]	Save the changes
C	Run the program to display the normal video character

This example produces the same results as a small assembly language program that written out would be:

```
LD A,8          ;Load A with 8
LD B,25         ;Load "Normal Video" character into register B
RST 8           ;Invoke SVC subroutine. This statement is the final
                ;statement in every usage of a SVC and behaves like
                ;CALL SVCnn where nn is the number in the A register,
                ;except that in the A register the number must be in
                ;Hex notation
RST 0           ;End of program—return to TRSDOS
```

If this assembly language program were compiled by the TRSDOS compiler M80 and loaded by the TRSDOS loader L80, the result would be the following string of hexadecimal codes, stored in the computer's memory:

3E 08 06 19 CF C7

3E 08	= Load Register A with 8
06 19	= Load Register B with 25 decimal
CF	= Call the SVC whose number is stored in Register A (specifically VDCHAR)
C7	= Return to TRSDOS

This program, when run, would perform the loading of the A and B registers which you accomplished manually via DEBUG. If the registers are loaded manually, the only part of this program needed is the "CF C7" part.

Here's another demonstration using SVC #15 called DISKID that will read your disk IDs into memory. Note the Entry Conditions for this SVC in *The Owner's Manual.*

DEBUG*

R	Another example of manually loading the registers
A0F	This accomplishes the same thing as an assembly language program: LD A,15 (=0F hex)
>> FF	This accomplishes LD B,255 (=FF hex)
>>>>>> 2810	And this does LD HL,2810. 2810 is the start of a 32-cell "buffer" area that will receive the data placed there by the SVC
[F2]	Save the register data

This time we will set a breakpoint in DEBUG to avoid slipping back into TRSDOS. This is done by:

B	Machine asks "# = .", i.e., "which of the ten breakpoints do you want to set?"
1	It asks "A = ...", i.e., "at what address should the #1 breakpoint be set?"
2801	In place of C7 at this address, D7 appears!
C	Now we run the program, up to the breakpoint

Presto! Your disk drive names appear in locations 2810-282F (with blanks for inactive drives). But you are still in DEBUG. The effect is the same as if you had written the assembler program in the second column below (ignore the first column for now):

```
06 FF          LD B,255

21 10 28       LD HL,x'2810'         ;hex 2810 is the start of the "buffer area"

3E 0F          LD A,15               ;15 = number of DISKID SVC

CF             RST 8                 ;call the SVC
```

The first three lines in the second column above establish the required entry conditions for the DISKID SVC when used to get the IDs for all four drives. The fourth line is the "call" to the 4SVC subroutine which places the names of your disks into the specified area in memory (2810-282F hex).

All SVCs can be run "manually" as I've shown in the above examples. Typically, a useful program consists of several SVCs with a good bit of other code besides. In the current example, we can display the DISKIDs which so far are stored in memory and have not been officially "displayed." Of course, you can read them, but that's because you are in DEBUG; if you had not set the breakpoint, you would have exited to TRSDOS and seen nothing displayed. The display can be accomplished by carefully typing in the following string of entries, shown all on one line to save space but with white spaces for readability:

```
E R A 09 >> 20 0D >>>> 2810 [F2] J 2800 C
```

The "E" gets rid of the breakpoint. The "J 2800 C" runs the program after setting the program counter to the start of the program. The stuff in between is the manual equivalent of the following program (again, ignore the codes at the left for now):

```
3E 09          LD A,9                ;VDLINE SVC number

06 20          LD B,32               ; = 20 hex = the number of chars to be
                                     displayed

0E 0D          LD C,0DH              ;"return" character at the end

21 10 28       LD (HL),x'2810'       ;2810 hex is where data is
```

and the program CFC7 in memory is

```
CF             RST 8                 ;call the SVC named in Register A

C7             RST 0                 ;Return to TRSDOS when done
```

Look up the VDLINE SVC in the *Owner's Manual* and see how the preceding program data fits the required entry conditions for this SVC which reads memory data and displays it on the screen.

Writing Programs Which Include SVCs

In simple cases, that's all there is to using SVCs — just load the registers and call the SVC. Of course, it's tedious to load registers manually, and it's easier to write the assembly language program shown above to do it. However, for quick jobs, or if you don't have the TRS assembler programs M80 and L80, you can do it with DEBUG. For example, what was done above for the DISKID SVC and the VDLINE SVC can be accomplished by:

```
DEBUG*

M 3000 [F1]                       ;Get ready to enter code starting at Hex 3000

06 FF 21 20 30 3E 0F CF           ;This is the DISKID code above (you ignored it) but modified so that the
                                  ;data is stored in locations 3020-303F hex

3E 09 06 20 0E 0D 21 20 30 CF C7  ;This is the VDLINE code similarly modified

[F2] J 3000 C                     ;This saves the program and runs it
```

The advantage of doing it this way is that now you can save the entire program permanently by typing at TRSDOS READY

```
DUMP DISKIDS (START=3000 END=303F)*
```

And you will then have a program on disk that you can use any time. First turn off DEBUG with

```
DEBUG OFF*
```

To use the new program, you can just type

```
DISKIDS*
```

Try it. Put in a different disk in Drive 1 (if you have one) and test the new program.

Now you can see what those "ignore for now" codes in the previous two examples were — the compiled, or machine language, version of the assembly language programs. To get these codes you have to consult a source on Z-80 programming (for example, *Z-80 Microprocessor Programming and Interfacing*, by Nichols, Nichols, and Rony, published by SAMS), but for simple jobs you can get by with this abbreviated table.

To create the necessary Entry Conditions for the SVCs:

WHEN IT SAYS	THE ASSEMBLY CODE IS	AND THE COMPILED CODE:
A = n	LD A, n	3E n'
B = n	LD B, n	06 n'
C = n	LD C, n	0E n'
D = n	LD D, n	16 n'
E = n	LD E, n	1E n'
(BC) = address	LD BC, address	01 address'
(DE) = address	LD DE, address	11 address'
(HL) = address	LD HL, address	21 address'

In this table, "n" stands for a number from 0 to 255; "n'" stands for the corresponding number in hex, from 00 to FF; "address" stands for an address in memory that in hex form is JKLM; and "address'" stands for the same address but with first and last parts inverted (thus, LMJK) according to the peculiar convention used by the Z-80. Using this table, you can hand-compile simple programs and enter them via DEBUG and save them via DUMP as we did above.

At this point, you should have enough information to experiment: try RAMDIR, which loads all your directory information into memory at a location you select by (HL) = address. (Hint: set a DEBUG breakpoint right after the "CF" or you will probably have trouble.) If you succeed, try putting the directory information into alphabetical order by filename, using the SVC SORT. Try printing out the date to your printer, if you have one, using the SVC DATE and SVC PRLINE.

An important thing to be aware of when constructing programs this way or when using the assembler: you should always assume that calling a SVC with RST 8 ("CF") destroys the contents of all the registers. Actually, RST 8 ("CF") doesn't always do this, but the *Owner's Manual* doesn't tell which registers are altered by the subroutine and which are not, except for those listed in the "Exit Conditions" information. And they're almost always altered. So play it safe.

Not all of the SVCs are quite as easy as the examples above. In fact, some like PARSER and OPEN have to be studied with care and require not only loading registers but also creating buffers (sections of memory set aside for storing data), preloading the buffers, and so on. You probably should stay away from complex ones like these unless you have the M80 assembler program or are a very careful hand-compiler.

For Assembly Language Programmers, Mostly

If you have M80, you can save yourself a lot of effort by creating a macro library, consisting of macros that do most of the dog work for you. I have a macro in my library MACLIB/MAC as follows:

```
VDLINE        MACRO         BUF             ;BUF is the symbolic name of the
                                            data
              LD            HL,BUF          ;buffer holding data to be displayed
              LD            B,L&BUF
              LD            C,0DH
              LD            A,9             ;VDLINE SVC
              RST           8
              CALL          NZ, ERROR
              ENDM
```

When I want to use this, I call it from with my Assembly program by writing the one line:

```
VDLINE OUTBF
```

where OUTBF is the name of some arbitrary data buffer, and the M80 assembler program in effect replaces this by the six lines which set up and call the SVC, as follows:

```
              LD            HL,OUTBF
              LD            B,LOUTBF
              LD            C,0DH
              LD            A,09
              RST           8
              CALL          NZ,ERROR
```

Of course with this one, somewhere in the data section of the program must appear

```
OUTBF:        DS            (some number of bytes to be reserved)
LOUTBF        EQU           $-OUTBF
```

Here the saving is not very spectacular, but with a biggie like OPEN or PARSER, it is very substantial indeed.

I start building any new assembly program by doing

```
COPY MACLIB/MAC NEWPROG/MAC*
```

where the latter is the name of the new program, and then I

```
EDIT NEWPROG/MAC*
```

to keep the SVC macros to be used and edit out the rest.

Errors (Heaven Forbid!)

One major item has been glossed over — the matter of errors. SVCs either do what you expect, or else they refuse and notify you of an error (for example,trying to VDCHAR a non-ASCII character is an error). If an error occurs, the error code shows up in Register A and the Z flag is NOT set, as you can see in the "Exit Conditions" section of most SVCs in the TRSDOS manual. The statement "NZ= error" means that if you test for that condition, you can take an appropriate branch.

It is dangerous not to check for errors, and it is good programming practice to check for the error indication after every SVC call and deal with the error if it arises. Typically, this is done by "CALL NZ,address" or "JP NZ, address" where "address" is the location of an error handling routine. When an error occurs, the error number is stored in register A so that you can tell what went wrong. The error handler I've developed tells me both where the error occurred and what went wrong. Incidentally, it also illustrates the use of macros and a few more SVC calls. It is:

```
ERRMAC      MACRO           ;insert after end of main program, as a
                            subroutine
            CSEG
ERROR:      LD B,A          ;get and save error message using ...
            LD HL,ERRMSG
            LD A,52         ;ERRMSG SVC
            RST 8
            EX (SP),HL      ;Deduce location of code which called
            PUSH HL         ;this error routine, and get it into
            DEC HL          ;register pair DE
            DEC HL
            DEC HL
            EX DE,HL        ;now DE holds the location of the call
            BINHEX ERRADD   ;convert location to ASCII via BINHEX
                            Macro
            VDLINE ERRBF    ;Display error message via VDLINE macro
            POP HL
            EX (SP),HL      ;Restore stack (actually unnecessary)
            RST 0           ;Kill the program, go to TRSDOS READY
            DSEG
EFFBF:      DB "Error called at hex location"
ERRADD:     DS 4            ;Location, in ASCII, is placed here
            DB 0DH          ;"Return" after printing message
ERRMSG:     DS 80           ;System error message appears here
LERRBF:     EQU $-ERRBF     ;Length of the entire message
            CSEG
            ENDM
```

This macro in turn uses two other SVCs — VDLINE (givenabove) and BINHEX.

```
BINHEX      MACRO HEXBF     ;converts hex number in DE to ASCII
            LD B,0          ;version in buffer HEXBF,using...
            LD HL,HEXBF
            LD A,24         ;BINHEX SVC
            RST 8
            CALL NZ,ERROR
            ENDM
```

Miscellany, For The Hard Core

When RST 8 is executed, the number in the A register is doubled and added to 01B8 hex to get the location of the SVC subroutine. For example, for SVC 36 (=24 hex) 2 times 24 plus 01B8 equals 0200 hex. At 0200 we find address 1350 hex. So the code for SVC 36 actually starts at 1350. If speed is important, it would be faster to write "CALL 1350H" rather than "LD A,36; RST 8".

The *Owner's Manual* indicates that you can add your own SVCs using the SETUSR routine. I've tried this and it works, but it seems like a practically useless feature since the resulting subroutine must be loaded before use; i.e., it is NOT made a permanent part of the boot-up system. You might as well put your original SVC into your macro library and build it into any program you write that uses it.

What you might find even more useful would be to install your compiled SVC code right into the system area via a patch to the system, but when adding such code, you would have to be sure it would be in a section of memory that would never be used for anything else. And that is tricky business unless you overwrite the code for an SVC that you are sure you will never use, even indirectly, since TRSDOS calls these routines also. That is a tricky business unless you have a thorough understanding of TRSDOS and how it uses the memory. You might substitute your code in place of the code for a built-in SVC, for example, if you are sure you will never want the built-in SVC. But you must also be sure that TRSDOS *itself* will never need the original code, and that is hard to determine since some SVCs call other SVCs and none of these complex dependencies are in the public TRSDOS documentation.

The SVC arrangement on the Model 16 is different, and I have no experience with it since my II-to-16 up-grade kit has not yet been installed. Many SVCs for the Model II are not given in the manual for the Model 16. Perhaps the missing ones will be added in a future release of the Model 16 software — but who knows.

THE RANDOM HOUSE ELECTRONIC THESAURUS FROM DICTRONICS PUBLISHING, INC.

"Reference Software for Microcomputers"

Barbara Albert

I can't wait to tell all of you about my latest acquisition, the RANDOM HOUSE ELECTRONIC THESAURUS from Dictronics Publishing, Inc. When it came in the mail, I felt like a kid finding an especially-wanted present under the tree at Christmas. This word-processing enhancement software is the most exciting hing I've seen in a long time.

JUDGING THE THESAURUS

Appearance

Where to start telling you about it? Perhaps the first thing I noticed after I stripped off the package's mailing wrappers was its appearance, so I'll start there. Usually with new software, there's nothing to notice or comment on so it was indeed a pleasure to find that the Random House Electronic Thesaurus comes in a nice, flat, eminently storable box (9" x 8-1/4" x 1/2") with a simple, subtle, attractive logo on the top. This packaging is such a sensible way to store two disks and their accompanying documentation! It's great. In addition to the disks and the documentation, Random House has included a pocket-size copy of the Thesaurus. That's in the box, too. I guess you can call that the "hard copy printout" of what is otherwise machinable media.

Documentation

The documentation is the next thing a ew software owner probably turns to after admiring (or not admiring) the packaging; I know I do. I cannot praise the documentation of The Random House Electronic Thesaurus highly enough. It's precise. It's concise. It's accurate. It's helpful. It's informative. It's explicit. It's clear. It's understandable. It's thorough. It's complete. It's effective. It's logical. It's everything documentation should be...and more.

It's slick and professional in its appearance and in its prose style. Although no credit is given to any writers, the Fifth Avenue, New York City address on the back of the booklet offers us a hint that this work is the very admirable result of a collaboration between people who know what they're doing, people who know how to write. There is no way you can have any questions about what is going to happen or how you make things happen with your Electronic Thesaurus after reading the documentation. It's perfect.

In addition to providing impeccable instructions for operating The Random House Electronic Thesaurus, the unknown authors supply all kinds of interesting facts about the development of the Electronic Thesaurus. FACT: It took more than five man-years to produce. FACT: It contains 60,000 synonyms in the standard 240K version. FACT: Ernest Freeman, a Dictronics programmer, developed a secret process for compacting 700,000 bytes of Thesaurus entries into 240K without reducing performance. FACT: When using the Electronic Thesaurus, the "average person" can look up and re-place a word in his text in less than ten seconds. FACT: The Electronic Thesaurus can perform so rapidly because it minimizes the use of the disk drive. The number of disk accesses is kept way down so the the speed can be kept way up.

Okay, so the packaging is attractive and practical, and the documentation is nothing less than perfect. Quit stalling and get to what everyone really wants to know. How does it perform? Are slick packaging and impressive documentation all this thing has going for it?

No. That is not all it has going for it, and it performs beautifully and easily.

Installation

The Electronic Thesaurus comes with four files: Modify.Com, the installation program; Teach.Rht, the Instruction Program; Words. Rht, the Thesaurus itself; and Rht.Com, the Thesaurus utility program. After making back-up copies of the original disks, place the installation disk in Drive 0 (the installation disk contains the three working files, not the Thesaurus itself). Place your WordStar disk into Drive 1. Log onto Drive 0; type Modify (Return) and then answer the questions as they appear on the screen. In no time at all, the Thesaurus program is up and running.

The installation instructions are embarrassingly simple, and the questions flashed on the screen during installation require virtually no explanation at all. I figured out how to do the entire installation all by myself, with no help other than the Manual. That attests to how easy it is; it also attests to how clearly the directions are written. If you do manage to get confused somehow, the Manual includes a three page series of bold-printed, green-outlined blocks which detail the installation process. You really can't go wrong.

Performance

After the Thesaurus is installed, you will no longer call WordStar at boot-up with a simple WS. Now it is WST — WordStar with Thesaurus appended thereto. (You can give it any other call letter you care to. The Manual suggests WST and refers to it as WST throughout. So if you give it another letter on the end, you have to remember what you called it.)

After the Thesaurus is in, you get a message that announces "The installation of The Random House Electronic Thesaurus is completed and OK." You're ready to try it out! Remove the installation disk from Drive 0 and replace it with the disk containing the actual Thesaurus.

You need remember only six little commands to use the Thesaurus, and they're real easy ones. Call up WST and open a file you're working on. Position the cursor on a word for which you would like to check synonyms. Press ESCAPE twice.

The screen will divide in half, and your text will appear on the lower portion of the screen. The upper half of the screen turns into the Thesaurus, with the word you want to check and all its synonyms listed right before your eyes.

Read over the various synonyms and select the one you want to use to replace the word in your text. Position the cursor on this word and press ESCAPE once. The Random House Electronic Thesaurus will AUTOMATICALLY replace the original word in your text with the synonym you selected! And in less than five seconds!! It is so neat; you'll want to find synonyms for and replace every word in your work. I was so excited when I got it, I showed it to everyone who came anywhere near me. (The fellow delivering water wasn't too interested, however.)

The rough approximation of the actual Random House Electronic Thesaurus display shown in Listing 1 gives you an idea of the type of thing you will see on your screen when you call up a synonym listing. Please note the menu. This always appears so there is never a time when you are wondering "what do I do next?"

Because disk space is limited, you will occasionally ask for a word that is not listed in the Thesaurus. When that happens, you will be shown the approximately 30 words which alphabetically surround the word you're looking for. Chances are, the root form of your word will be in this list. If you position the cursor on the root form of your word and press ESCAPE once, you'll probably find a usable synonym. You'll have to change the noun root word into an adjective form or a present tense verb into past tense or whatever it is you need, but that isn't too difficult. (See Listing 2.)

If you call for the synonyms of a word and don't find one you want to use, all you have to do is press (RETURN) and the Electronic Thesaurus will return control to you, and the cursor will be at the position you left it.

The Electronic Thesaurus can also be used from the command level of CP/M using the RHT.COM Utility Program. Of course, all you will get when using it this way is a list of synonyms; you can't replace any words in your work. But if one of your co-workers asks you for a synonym, you can show off a little with your Electronic Thesaurus. You can probably look up ten words before the other person even locates the hard cover Thesaurus on the book shelf.

Recommendation

In conclusion, buy the Random House Electronic Thesaurus if you ever have occasion to look up synonyms. For anyone who has ever created a written document using word processing, this is a must!

Ethical Considerations

To say that anything is perfect really goes against my better judgment — I mean, NOTHING is perfect, right? Since I feel I must find something to criticize about the Electronic Thesaurus, I will say that it bothers me because it makes me lazy Here's one more electronic gizmo in my life doing one more thing for me that my brain used to do. I currently rely so heavily on PROOFREADER when I'm writing that I tend to be a little careless about spelling. Now with the Electronic Thesaurus, I find myself becoming complacent about word choice. I put down any old word and tell myself I can always substitute better words during editing with the Electronic Thesaurus. Is that bad/wrong? I don't know...but I do know that until more evidence is in concerning the ethical implications of allowing the computer to do your thinking for you, I'm going to continue consulting the Electronic Thesaurus as often as I can.

RANDOM HOUSE ELECTRONIC THESAURUS
is available from
Aspen Software Company
P. O. Box 367
Tijeras, NM 87059
Price: $150.00
Available for CP/M systems ONLY
Call 505/281-1634 for more information

```
BARE - (adj.)  naked, nude, unclothed, undressed, exposed,
unprotected, unsheltered, unshielded, open, stark, unadorned,
barren, simple, sheer, mere, alone, undisguised, obvious,
glaring, evident, palpable.  (v.)  disclose, denude, lay
open, expose.

!<ESC> TO CHANGE        !<SPACE BAR> TO MOVE WORD RIGHT!
!<RETURN> TO EXIT       !<BACKSPACE> TO MOVE WORD LEFT !
!<CTRL-W> TO SCROLL UP!<CONTROL-Z> TO SCROLL DOWN      !


------------The Random House Electronic Thesaurus------------
------------                                     ------------
```

Listing 1

This is a representation of the top half of your CRT when the Random House Electronic Thesaurus listing of the word "bare" is invoked. Note the groupings of the synonyms in sets of submeanings.' The Manual states: "Where a listing of synonyms all reflect a similar submeaning, we have sometimes attempted to group the words in order of their 'power' of meaning." No division between submeanings has been included to conserve space.

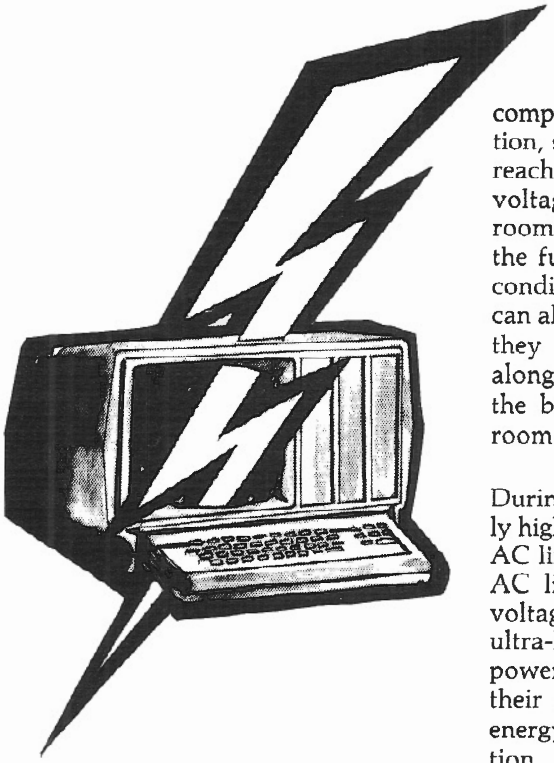
```
Requesting:  teaching

teaching IS NOT LISTED.  THESE WORDS ARE ALPHABETICALLY
CLOSEST:

target, tarnish, tarry, task, taste, tasteless, taunt,
tavern, tawdry, teach, teacher, team, tear, tease, technical,
tedious, teem, tell, temper, temperament, temperamental,
temperate, tempestuous, temporary, tempt, tempting

          !<ESC> TO SELECT A WORD AND LOOK IT UP      !
          !<RETURN> TO EXIT AND RETURN TO YOUR TEXT !
          !<SPACE BAR> TO  MOVE WORD RIGHT            !
          !<BACKSPACE> TO MOVE WORD LEFT              !
```

Listing 2

This is the screen display you would get if you asked for the word "teaching." As you can see, even though "teaching" is not there, "teach" is, so you position the cursor on "teach" and press <ESCAPE>. When the entry for "teach" appears, you can proceed as with a normal entry.

MORE POWER TO YOU

John Thornton
International Software Sales, Inc.
P. O. Box 223
Newtonville, NY 12128



As computerists, our dependence on the local power company and the electricity it generates is one common bond that links us together. It is also, perhaps, our greatest weakness. We may be forerunners for the future, declaring our independence from traditional and outmoded ways of thinking and doing business. But when it comes to electricity-dependence, we are as powerless and vulnerable as the guy next door who doesn't know the difference between a TRS-80 Model II and the Atari his kids got for Christmas. Our computers MUST have the electricity produced the power companies to operate. Unfortunately, this essential electricity can also be a villain that generates many problems for you and your computer.

What Power Line Changes Mean To You
Like most of you, I, too, have been a victim of the mysterious and irritating problems that power-line transients (voltage surges/spikes) and noise (or hash) can create with micro-computers. These problems present themselves in many different forms, and the only similarity in the way in which power related disturbances manifest themselves is the computer's erratic and, to the unknowing observer, unexplainable behavior.

For example, the computer might read one drive and then the other repeatedly for several hours. Or the cursor will present itself on the screen right in the middle of your program. Or you try a few random keys on the keyboard, and get no response. You hit BREAK. Still no response. Looking up at the RESET button, your heart stops at the thought that all your data is somewhere inside there, and you don't know how to get it out.

This type of problem, along with many others, may be caused by sudden changes in the line voltage or other interference on the power source. A line spike, causing the line voltage to jump above 110 volts for even a millisecond, can result in loss of computer memory. Alteration of computer memory is the most common effect of voltage changes on computers. If you are trying to store information on disk during one of these spikes, you could end up with a flawed disk directory. Your computer is highly susceptible to electrical interference because it operates on extremely low power levels. Whenever even the smallest voltage changes assault its vulnerable and sensitive circuitry, the computer gets confused since the voltage interference on the line and the signals it receives when data is being input from the keyboard are practically identical. In other words, the computer can interpret power line noise as LEGITIMATE INPUT SIGNALS!! You can see why your computer's memory is in jeopardy from power line fluctuations!

Power line surges or spikes are brought about by lightning (the most frequent and most destructive cause), heavy machinery, restart after power outages, and power load switching by electric companies. During heavy machine operation, significant surges can be created that reach two or three times the normal line voltage. The copy machine in the next room, or the elevator down the hall, or the furnace in the basement, or the air conditioning unit outside your window can all cause surges on the line when ever they start up. And these surges speed along the electrical wiring to all parts of the building, including your computer room.

During a surge, short rushes of extremely high voltage are developed between the AC lines, and anything connected to an AC line receives a shot of super-high voltage. Microcomputers operate on ultra-low voltages and cannot tolerate power deviations. Consequently, one of their greatest risks is posed by high-energy overvoltages of even short duration, and by "short duration," I mean millionths of a second — "micro-seconds" — or billionths of a second — "nanoseconds"! They are extremely sensitive to even the very lowest levels of voltage surges.

Types of primary damage caused to electrical components and integrated circuits by a direct hit from a surge include false triggering of circuits, hazardous shorts, and even total burn-out or destruction of electrical components (in some cases, the voltage may be high enough to arc across "OFF" switch contacts and cause an electrical fire!)

Secondary damage includes reduced performance or shortened life spans in electrical components; they begin to fail at a more rapid rate than normal and "glitches" begin to appear in systems which had previously been operating flawlessly.

Power line static, called "noise" or "hash", consists of unwanted signals on the power lines and causes electrical pollution that is good for absolutely nothing but interfering with your computer's operation. Hash is created by hundreds of sources, including arcing in different electrical tools, appliances, and microprocessors; defective, loose, or corroded wiring in light sockets, wall sockets, line-cord plugs, and wire connections; interaction between computer disk drives, printers, and other peripherals; and power lines which act as antennas for unwanted interference when they pick up various signals (CB sets, police and fire radio transmissions, etc.).

Externally created hash interferes with smooth microcomputer operation: "glitches" appear in programs which ran perfectly before; erroneous printouts show up; erratic computer/peripheral behavior occurs.

The following list shows a sampling of symptoms your computer may exhibit when exposed to line voltage changes:

- Unaccountable Read/Write Errors (The first sign of trouble)*
- Loss of Memory (in BASIC or TRSDOS)
- False Error Messages
- Printer/Computer Lockup (No response to input from keyboard)
- Computer Resets by Itself
- "Glitches" in previously perfect programs
- Random Characters Displayed on the Screen
- Shrinking Video Display
- Misindexed Programs
- Uncontrollable Printer Behavior

What Is Tandy's Solution

Radio Shack's TRS-80 Model II Owner's Manual states: "Your TRS-80 Model II contains a specially designed, built-in AC line filter. It should eliminate all but the most severe interference problems."

In my experience, I have found that — in most cases — this built-in AC line filter is NOT able to control line voltage.

The built-in filter does not offer any protection against major power transients. It does not offer any protection against lightning strikes of utility wires. (This is especially frightening since the strike could be a mile away and still do major damage.) It also does not offer any protection against certain switching relays the power company may use.

What Radio Shack's built-in line filter does is filter out some of the noise or hash. The line filter eliminates 70 to 80 percent of the power line noise that is making its way into your computer via the wall socket. But your computer is only BUFFERED from damaging effects, not completely isolated. Twenty to thirty percent of the damaging noise on the power lines is causing destruction to your disk drives, the first place trouble will show up.

I have seen two pieces of computer equipment, both directly connected to wall outlets, severely damaged during electrical storms. Both computers required several hundred dollars of repair. Both owners believed they were protected by the built-in AC line filter.

Radio Shack does not sell or offer advice concerning the purchase of any type of power supply protection. A 110-volt grounded line, as long as it's a "clean line" (not being used for any other equipment or appliances), together with the built-in AC line filter, offers protection for your computer against power spikes and surges, electrical storms, and relay switch ings that is "good enough," according to the manager of one Radio Shack Computer Center.

How Can You Protect Your Computer?

I recommend to my clients that all computer installations be equipped with a true line voltage regulator/transformer. A 400 watt unit — adequate for operating a Model II or 16, one expansion drive, and a printer — costs approximately $600. (A computer system with a hard drive will require slightly more power protection — at least 600 watts.) The Sola Regulator is the best possible unit to buy; it will protect against short-term voltage drops and other voltage transients on typical household circuits by automatically increasing or reducing voltage to compensate for incoming voltage variations. You can calculate a rough estimate of the cost for this type of protection by figuring on $1.50—$2.00 per watt.

If a client feels the price of a voltage regulator is too high, I urge him to get an isolation device, preferably an ISOLATOR from Electronic Specialists in Natick, Massachusetts. The ISOLATOR line of products provides heavy duty surge/spike suppression, hash filtering benefits, and filtered isolation between inter-acting equipment. This last type of protection is needed because many computer systems create their own electrical pollution. When disk drives, printers, and other peripherals interact, they create enough interference to disrupt efficient computer operation by causing any number of problems, such as those in the list above. Isolators can also be incorporated into a total computer system protection plan by installing them where wall or light socket connections are loose or worn. These seemingly innocent sockets very often cause line interference in the same room as the computer.

An excellent alternative to the line voltage regulator and/or isolation device is a motor generator set. This set provides either an AC or DC generator for the entire computer system that will generate a pure 110 volts, free from all power company line noise. It also provides capacitors and inductors that have the strength needed to absorb surges and spikes that come on line. Unfortunately, this is not an inexpensive solution to the power protection problem.

A Surge Sentry® from RKS Industries, Inc. in Scotts Valley, California, constantly monitors incoming voltage for transient surges. When any voltage rise of more than 10 percent is detected, the unit responds in picoseconds by "clamping" the line. The dissipation rate for the Surge Sentry® has been measured at 1,000,000 watts at 100 microseconds. This type of protection allows your computer to continue operating during a power surge. Costs range from $70 to $150 for each unit.

An uninterruptible power source (UPS), such as the May Day from Sun Research, Inc., in New Durham, New Hampshire, will provide the ultimate protection against the very destructive power line transients. This unit comes equipped wit a battery pack to insure a constant, even supply of power to your computer in the event of a brown-out or even a total power outage. The UPS is designed to maintain a program in the computer

*Note: Some write errors will remain undetected until the next processing of the data file. This could result in the backup of data that is worthless or incomplete.

a program in the computer when the power drops or shuts off; it allows the operator to save the program on disk by performing normal shut down procedures. No data lost! No programs crashed! The length of time the battery will last depends on how much you wish to invest in this type of unit. Battery time will average from about three minutes for the less expensive units to about half an hour for the units in the $1,000+ range. May Day products promise you a minimum of 30 minutes.

Caveat Emptor

BEWARE: Many units advertised as power line protection devices will do little or nothing to correct voltage problems. If you are planning to spend money on a regulator, be sure it is a line voltage regulator and not a "Line Conditioner" or "Line Stabilizer." These units CANNOT regulate line voltage and may prove to be a waste of money.

I was recently given a "Line Stabilizer" to test by a Model II owner. This unit had been sold to him by a salesman who promised it would eliminate any and all line problems he could possibly encounter and provide complete power protection for his microcomputer. The manufacturer's literature claimed this 1500 watt line stabilizer is "Ideal for: powering mini computer systems and computer perifherals (sic); operating sensitive test apparatus; powering refrigerators and air conditioners; powering television sets; wherever constant AC line power is required."

Since I'm a disbeliever, I found it difficult to understand how a unit that retails for $125 could be suited for use on a micro. So I called the manufacturer. After reading from the sales literature supplied with the unit, I asked the manufacturer's spokesman if he could justify his company's claim that the unit is suited for use on mini computers. He replied that they never really claimed to sell a regulator, only a line conditioner.

I found this highly unlikely as I read directly to him from their literature: "Ideal for powering mini computer systems..." He finally admitted that the unit had been developed solely to supply constant power to refrigerators and other household appliances. They added the part about computers to their advertisement when someone installed one of their units on a computer and reported that "it helped." After such resounding success with minicomputers, their salesmen were authorized to sell the units for microcomputers, too.

Miscellany

The TRS-80 Model II Owner's Manual offers some wise advice: "The more complex the system and the more serious the application, the more consideration you should give to providing an ideal power-source for your Computer." How much is your data worth to you? If you can afford to take chances, then go ahead. But if you can't, a little money spent on "power insurance" can go a long way.

RULE: Don't buy a unit only because it's cheap. Estimate how much your data and your time are worth. Remember, you get what you pay for.

How do you decide which power protection device is right for your system? Ask the salesman for names of people who have purchased the device you are considering, and then talk to the actual users. Check and double check the credentials of the manufacturer and the product before you lay down some hard-earned cash. (How long have they been marketing power protection for computers? Do they consider power protection for computers seriously, or is it only a side line for them? Some literature I have seen discounts the danger of power line transients, but only because that particular manufacturer doesn't sell a line-filtering product. Others, such as the one I discussed above, decide to jump on the "computer bandwagon" and announce that a product of theirs — designed to supply power protection to a TV set or a stereo — is now also capable of protecting a computer.This type of product is probably not for you.) Ask the opinion of someone you trust. If you don't know if your home or office is in an area subject to high levels of electricity pollution, call an electrician and ask him about the frequency of problems on the lines. Be suspicious of prices that are too low. This kind of protection doesn't cheap.

II/16

A PROGRAM THAT'S "OFF AND RUNNING"

HANDICAP THE THOROUGHBREDS!

A program for Radio Shack's TRS-80 Model Two/Sixteen that classifys and ranks thoroughbred flats based on scientific methods. **Handicapping**© will give you the edge and build skills resulting in rewards at the track.

The **Handicapping** system was developed by Kenneth M. Johnson, Ph.D., and Lawrence G. Archer, Ph.D., after much research and statistical evaluation. Based upon three important handicapping factors, it avoids the "hunches" and "opinions" that make up most handicapping systems.

Send for the **Handicapping**© program today . . . and let the thoroughbreds run for you tomorrow..

900-0006	TRSDOS 2.0	$100.00
901-0006	TRSDOS 4.1	$100.00
902-0006	TRSDOS 1.3	$100.00

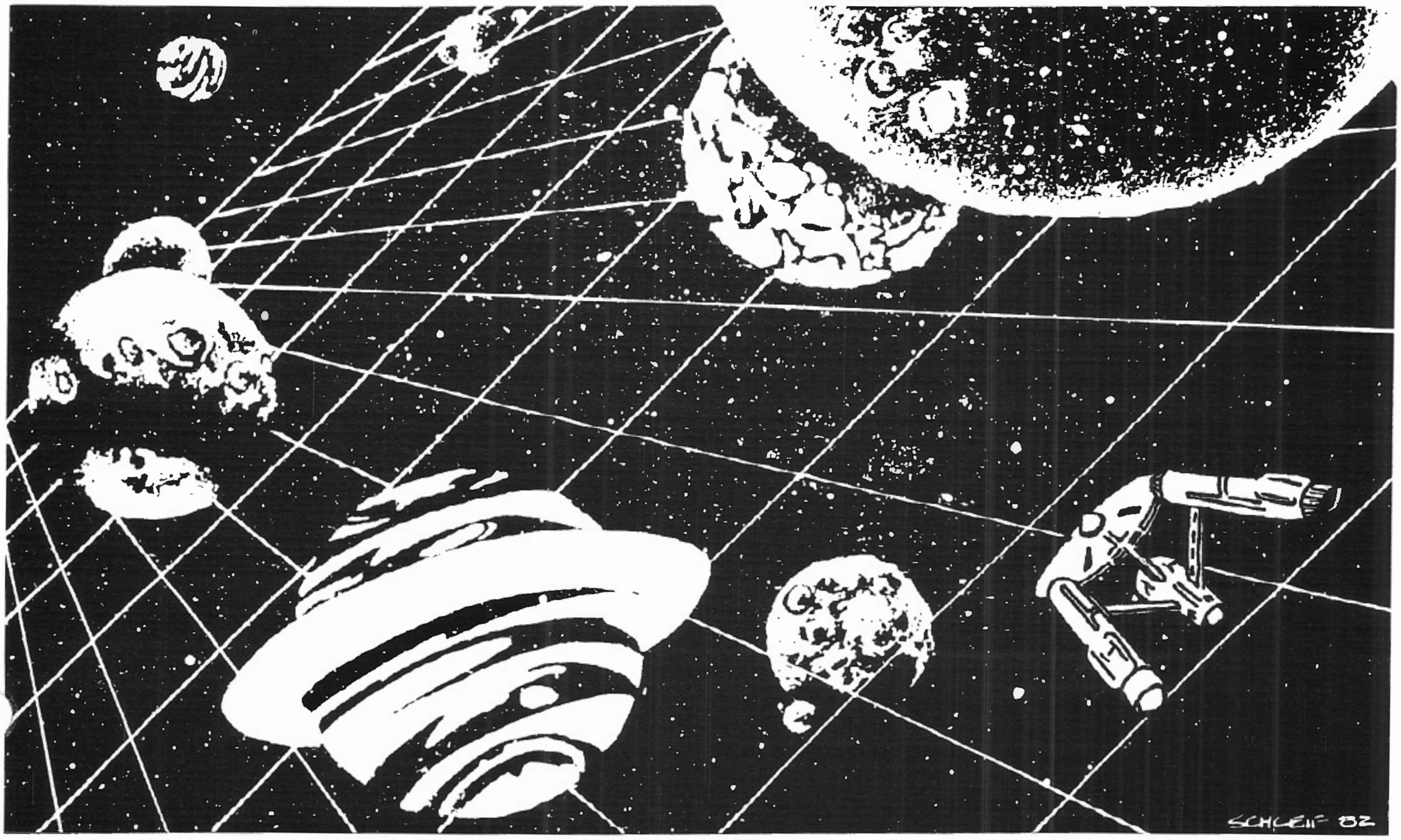
Rizzo Data Systems Corp. offers a full line of leisure programs for the Model Two/Sixteen. Call or write for a free brochure and current information.

Shipping Charge $3.00 Per Order - C.O.D.'s - Add $2.00
30 Day Money Back Guarantee On All Programs
Visa, MasterCard, Personal Checks or Money Orders Accepted

Rizzo Data Systems Corp.

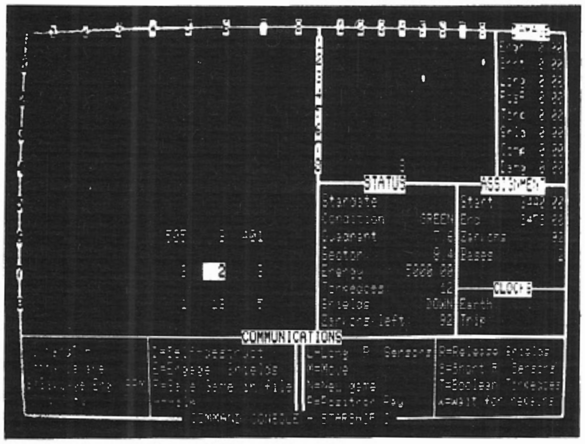
33 Westwood Ave P.O. Box 458 Bridgeton N.J. 08302 0356 • 609 451 7964

STARSHIP I



STARSHIP/I (copyright registered) is a save-the-galaxy type game based on the venerable STARTREK games which have been played on large mainframes for more than ten years. It is specifically designed for the RS Model II and features a constant console display with no scrolling. Each game begins in a randomly arranged galaxy of 64 "quadrants," each containing 64 "sectors."

STARSHIP/I is a "strategic" game (as opposed to "tactical" games which involve reflex action but little thought). However, STARSHIP requires *quick* thinking, as all events are timed. Because STARSHIP is written in the RS BASIC interpreter language, users can modify the source code or use sections of the code as models to learn techniques which can be used to produce games of their own design.



starship I

STARSHIP comes with a disk instruction file which can be listed on your printer. It can also be listed on the screen at the beginning of a session. In addition, STARSHIP contains a few undocumented "surprises" at advanced levels of play (there are ten levels of difficulty). These surprises may be analyzed by reviewing the source code, then modified or deactivated if desired.

Games may be saved in files at any point for later play or for "duplicate" games in which several players start with the same layout and position and compete for the highest ending score.

STARSHIP/I was sold for several months at $50 through leading computer magazines. It is now available to **two/sixteen** subscribers at the special price of $30, including air mail (anywhere) in protective Kangaroo mailer.

COMPUTER FURNITURE BY Data-MATE®

INTRODUCTION

DATA-MATE® computer furniture is carefully designed and engineered to provide higher efficiency and productivity. It is far more than simply a support for your computer equipment.

Rugged steel construction is designed for durability. Cable and paper management systems provide efficiency *and* neat, uncluttered appearance. All equipment bays have ample ventilation for heat dissipation. Clean lines blend perfectly with hardware design and function.

Data-MATE® products are in stock for immediate shipment.

Operator comfort is becoming increasingly more important. Features such as height adjustments to reduce operator fatigue and casters for mobility and versatility have been designed into the Data-MATE line. And that means greater efficiency and higher productivity.

Catalog, prices, and further information available from

two/sixteen magazine (717)397-3364

Excellence in computer furniture technology.